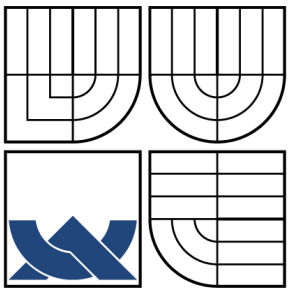
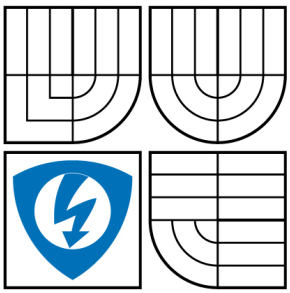


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VÝVOJOVÉ TRENDY PROTOKOLU TCP PRO VYSOKORYCHLOSTNÍ SÍTĚ

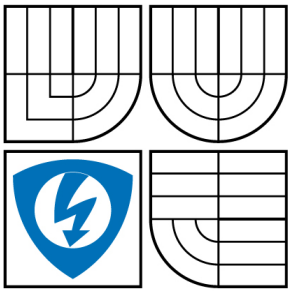
DEVELOPMENT TRENDS OF TCP FOR HIGH-SPEED NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JAN MODLITBA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JOSEF VYORAL
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor

Telekomunikační a informační technika

Student:	Modlitba Jan Bc.	***ID:***	47334
Ročník:	2	***Akademický rok:***	2007/2008

NÁZEV TÉMATU:

Vývojové trendy protokolu TCP pro vysokorychlostní sítě

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte princip spojově orientovaného protokolu TCP. Především se soustřeďte na modifikace tohoto protokolu(Highspeed TCP, Scalable TCP apod.), které jsou určené pro lepší využití přenosové rychlosti vysokorychlostních sítí. Výkonnost vybraných modifikací tohoto protokolu ověřte v prostředí simulátoru NS2 na k tomuto účelu vytvořených pokusných sítích. Na základě dosažených výsledků zhodnoťte vhodnost nasazení těchto protokolů do vysokorychlostních sítí. Nastiňte problematiku implementace do současných datových sítí.

DOPORUČENÁ LITERATURA:

[1] STALLINGS, W.: High-speed networks and internets Performance and Quality of Service, 2. vydání, Prentice Hall, New Yersey, 2002.
[2] Dostálek L., Kabelová A.: Velký průvodce protokoly TCP/IP. ISBN 80-7226-323-4, Computer Press, Praha, 2000.

Termín zadání:	11.2.2008	***Termín odevzdání:***	28.5.2008

Vedoucí práce: Ing. Josef Vyoral

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Bc. Jan Modlitba
Bytem: Palachova 1771/62, 59101, Žďár nad Sázavou - Žďár nad Sázavou 6
Narozen/a (datum a místo): 19.1.1983, Nové Město na Moravě

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 60200 Brno 2
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Kamil Vrba, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☑ diplomová práce
   - ☐ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: Vývojové trendy protokolu TCP pro vysokorychlostní sítě
Vedoucí/školitel VŠKP: Ing. Josef Vyoral
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 1
- ☑ elektronické formě - počet exemplářů 1

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☑ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

ANOTACE

Diplomová práce řeší problematiku nasazení nových TCP variant pro vysokorychlostní IP sítě. Cílem bylo nejdříve detailně popsat chování TCP a analyzovat problémy nevyužití dostupné šířky pásma linky standardního TCP ve vysokorychlostních sítích.

Práce se následně zabývá výběrem a popisem těch nejperspektivnější variant. Dále je čtenář seznámen s dostupnými síťovými simulačními nástroji dané problematiky a jejich stručným popisem.

Hlavní část práce představuje ověření výkonnosti TCP variant pro vysokorychlostní sítě. Při testování je brán ohled jak na výkonnost, tak i na férovost při soutěžení více přenosů na sdílené lince. Výsledky jsou přehledně zobrazeny a srovnány mezi sebou.

Klíčová slova - TCP, Kontrola zahlcení, Vysokorychlostní sítě, NS2, Simulace

ABSTRACT

The master's thesis solve the problem of setting new TCP variants for high-speed IP networks. The first goal was to describe in detail the behaviour of TCP and then analyse a problem of utilization the available bandwidth with standard TCP in high-speed network.

Work consequently deals with selection and description the most perspective ones. Further the reader is familiarized with reasonable simulation tools of existing problems and their brief description.

Main part of this thesis presents examination of performance of new TCP variants for high-speed network. During the examination the aspects on efficiency and fairness of competition flows on shared bottleneck are taken. The results are tabularly displayed plus compared with each other.

Keywords: TCP, Congestion control, High speed network, NS2, Simulation

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Vývojové trendy TCP protokolu pro vysokorychlostní sítě“ jsem vypracoval samostatně pod vedením svého vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne …… ……..

(podpis autora)

PODĚKOVÁNÍ

Tímto bych rád poděkoval vedoucímu diplomové práce panu Ing. Josefu Vyoralovi za ochotu, vstřícné jednání a za užitečnou metodickou pomoc.

V Brně dne …… ……..

(podpis autora)

SEZNAM ZKRATEK

TCP - Transmission control protocol

IP - Internet protocol

OWIN - Outstanding window

RWND - Receiver windoe

CWND - Congestion window

OSI - Open systems interconnection

UDP - User datagram ptotocol

MSS - Maximum segment size

MTU - Maximum transfer unit

IEEE - Institute of Electrical and Electronics Engineers

FDDI - Fiber distributed data interface

PPP/SLIP - Point-to-Point Protocol/Serial Line Internet Protocol

ISN - Initial sequence number

MD5 - Message-Digest algorithm 5

SN - Sequence number

FTP - File transfer protocol

RFC - Requests for Comments

RTT - Round trip time

RTO - Retransmit timeout

SACK - Selective acknowlwdgement

AIMD - Additive increase and multiplicative decrease

SSTHRESH - Slow-start threshold

BDP - Bandwidth delay product

ECN - Expilcit congestiob notification

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 ÚVOD

Více než 95 % objemu dat je dnes v Internetu přenášeno protokolem TCP. Tento protokol byl navržen v době, kdy sítě pracovaly na mnohem nižších rychlostech než nyní. V současné době koncové stanice a dokonce i laptopy jsou běžně vybaveny síťovými kartami typu Gigabit Ethernet, disponují gigabitovými pamětmi a daty, které mohou být přenášeny rychlostmi gigabity/s mezi pamětí a síťovým rozhraním. Současné IP sítě jsou vybudované s použitím multigigabitových okruhů a výkonnými přepínači a směrovači. Jestliže koncový uzel a stejně tak síť mohou oba podporovat gigabitové přenosy, pak i spojení mezi koncovými uzly s použitím protokolu TCP by mělo být schopné přenášet data v rychlostech gigabit/s stejně tak, jak dosahuje v megabitových sítích rychlostí megabit/s. Bohužel se však ukazuje, že použití protokolu TCP v rozlehlých vysokorychlostních a také bezdrátových sítích s sebou přináší určité problémy

Vysílač protokolu TCP musí regulovat rychlost posílání segmentů tak, aby nedošlo k přeplnění vyrovnávací paměti na straně přijímače nebo v některém ze směrovačů na cestě. Regulace se provádí omezováním množství odeslaných a dosud nepotvrzených dat, která mohou být v daném okamžiku na trase (*owin*). Mechanismus řízení datového toku se stará o přizpůsobení rychlosti posílání segmentů rychlosti přijímače. Přijímač průběžně informuje vysílač o zbývajícím volném místě ve své vyrovnávací paměti (*rwnd*). Musí platit *owin*<=*rwnd*. Vysílač podle různých typů signálů o blížícím se či již nastalém zahlcení sítě také vypočítává limit pro velikost okénka zahlcení (*cwnd*). Musí platit *owin*<=*cwnd*.

Standardní velikost vyrovnávací paměti pro jednotlivé TCP spojení je v současných operačních systémech nastaven obvykle na 64 kB, a to jak na straně vysílače, tak na straně přijímače. Této velikosti také odpovídá maximální velikost okénka během spojení. Pro rozlehlé vysokorychlostní sítě, mající velikou kapacitu dat, která mohou být v kterémkoliv okamžiku na trase, je tato velikost vyrovnávací paměti nedostatečná a je potřeba ji zvětšit. Zvýšení velikosti vyrovnávací paměti na kapacitu trasy lze provést třemi způsoby: ručně pro jedno TCP spojení na úrovni aplikace, ručně pro všechna TCP spojení na úrovní operačního systému a automaticky pro jednotlivá TCP spojení.

Bohužel zvětšení vyrovnávacích pamětí a tím i okénka vysílače nestačí k zajištění spolehlivé vysoké propustnosti v rozlehlých vysokorychlostních sítích. Je potřeba jednak upravit algoritmus řízení zahlcení na vysílači a jednak analyzovat a vyřešit celou řadu jevů, ke kterým při vysokorychlostních přenosech dochází.

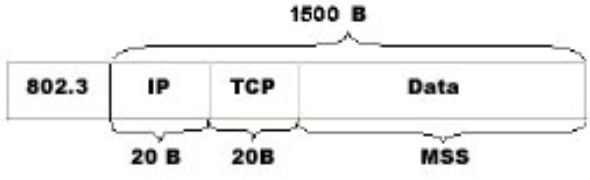
2 ZÁKLADNÍ PRINCIP TCP

Protokol TCP je součástí síťového modelu TCP/IP, který rozeznává pouze 4 vrstvy dle OSI modelu. TCP/IP zahrnuje síťovou (IP) a transportní (TCP, UDP) vrstvu, nejvyšší aplikační vrstva představuje určitou aplikaci, které pro svůj přenos využívají protokoly TCP nebo UDP a nejnižší vrstva síťového rozhraní umožňuje přístup k fyzickému přenosovému médiu.

TCP/IP je založen na principu datagramové služby přenosu dat sítí, kdy blok dat je rozdělen do menších částí – paketů a IP protokol zajišťuje přenos sítí pro protokoly vyšších vrstev. IP je nespolehlivý protokol, IP datagram nezajistí kontrolu přenášených dat, zabezpečí vyšší vrstvě pouze směrování datagramů v síti.

TCP vytváří dvoubodové duplexní spojení, před vlastním vysíláním uživatelských dat je vytvořené spojení mezi vysílačem a přijímačem, tzv. virtuální okruh. Spojení mezi aplikací odesílatele a aplikací příjemce je jednoznačně určeno IP-adresou, číslem portu a použitým transportním protokolem, číslo portu může nabývat hodnot 0 až 65535. Přicházející aplikační data jsou rozděleny na TCP segmenty, vkládají se do IP datagramu, který se dále vkládá do linkového rámce. Maximální velikost TCP segmentu (MSS) definuje největší množství aplikačních dat, které budou přeneseny jako jeden TCP segment. Pokud se použije příliš velký TCP segment, který se vloží do velkého IP datagramu, který je větší než maximální velikost přenášeného rámce (MTU), pak IP protokol musí provádět fragmentaci IP datagramu. Výběr správné hodnoty MSS je důležitý. Pokud je tato hodnota malá, využití sítě může být nízké a naopak pokud je tato hodnota velká, sníží výkonnost důsledkem vytváření příliš mnoha IP datagramů, které nemohou být samostatně ihned potvrzeny nebo přeposlány. Typicky je hodnota MSS nastavena na MTU mínus velikost TCP a IP hlavičky.

Síťový standard	MTU [B]
Ethernet	1500
IEEE 802.3/802.2	1492
FDDI	4352
X.25	576
PPP/SLIP	296
Token ring	17914



Tab. 2.1 - Velikost MTU pro vybrané síťové standardy Obr. 2.1 - Nastavení MSS pro Ethernet

2.1 HLAVIČKA TCP PROTOKOLU

TCP segment se skládá ze dvou částí:

- Hlavička
- Data

Hlavička obsahuje 11 polí, z nichž je 10 vždy nutně vyžadováno. 11. pole je volitelné

Bit offset	Bits 0–3	4–7	8–15	16–31
0	Source port			Destination port
32	Sequence number			
64	Acknowledgment number			
96	Data offset	Reserved	CWR ECE URG ACK PSH RST SYN FIN	Window
128	Checksum			Urgent pointer
160	Options (optional)			
160/192+	Data			

Obr. 2.2 - Struktura hlavičky TCP protokolu

1. **Source port** (16 bitů) - port procesu generujícího segment
2. **Destination port** (16 bitů)- určuje kterému procesu na cílovém uzlu jsou data určena
3. **Sequence Number** (32 bitů) - sekvenční číslo prvního oktetu v segmentu (pokud není nastaven příznak SYN). Pokud je nastaven příznak SYN, jedná se o tzv. initial sequence number – ISN a první datový oktet má číslo ISN + 1
4. **Acknowledgement Number** (32 bitů)- má význam pouze když je nastaven kontrolní bit ACK. Toto číslo je nastaveno na hodnotu, kterou odesílatel očekává v poli Sequence Number v následujícím segmentu. Je-li ustaveno spojení, je toto číslo vždy posíláno.
5. **Data Offset** (32 bitů)- specifikuje velikost TCP hlavičky. Minimální velikost hlavičky je 20B a maximální 60B. Indikuje, kde začínají být přenášena data v segmentu.
6. **Reserved** (4 bity)- vyhrazeno pro budoucí použití a mělo by být vždy nulové
7. **Flags** (8 bitů) - kontrolní bity (příznaky) zajišťující "handshaking" a ostatní specifické procesy
   - CWR (1 bit) – Redukce okénka zahlcení – příznak, který je nastavený vysílajícím procesem při přijetí TCP segmentu s příznakem ECE (podle RFC 3168)
   - ECE (ECN echo) (1 bit) – Během 3 – fázové inicializace spojení (Handshake) oznamuje, že TCP spojení je kompatibilní s ECN (RFC 3168)
   - URG (1 bit) - Oznamuje, že pole pro Urgent Pointer je platné
   - ACK (1 bit) - Oznamuje, že pole pro potvrzení ACK je platné
   - PSH (1 bit) - Push funkce

- RST (1 bit) - Reset spojení
- SYN (1 bit) - synchronizace sekvenčních čísel
- FIN (1 bit) - oznámení, že odesílající nemá žádná další data

8. **Window** (16 bitů) - množství dat v oktetech, které je potvrzováno najednou
9. **Checksum** (16bitů) - kontrolní součet, není povinný a v tom případě je 0
10. **Urgent Pointer** (16 bitů) - údaj je platný, pouze pokud je nastaven příznak URG
11. **Options** (volitelné množství bitů) - slouží pro různé doplňkové funkce. V současné době je jich definováno celkem 26, nejběžnější z nich jsou definované níže. Pokud se použijí, je hlavička TCP delší než 20 bajtů.
    - **Maximum Segment Size** - Explicitně určuje maximální velikost TCP segmentu
    - **Window Scale** - Umožňuje použít větší klouzavé okno pro řízení toku dat. Tuto volbu smí každá strana použít jen v úvodním SYN segmentu
    - **Selective Acknowledgement** - Podrobné informování odesílatele, které segmenty byly pořádku přijaty
    - **MD5 Signature** - Umožňuje připojit k segmentu kryptografický podpis založený na rozptylovací funkci MD5

2.2 ZAJIŠTĚNÍ SPOLEHLIVOSTI TCP

Protokol TCP zajišťuje aplikační vrstvě spolehlivý přenos dat, to znamená, že se data zaručeně přenesou k příjemci a pokud to není možné, oznámí toto aplikační vrstvě. Spolehlivost doručení aplikačních dat zajišťuje TCP potvrzováním přijatých segmentů. Vysílací strana rozdělí přicházející aplikační data do segmentů a každý opatří pořadovým číslem SN (Sequence number). Pořadové číslo SN je pořadové číslo odesílaného prvního bajtu TCP segmentu. TCP segment tedy nese data od pořadového čísla odesílaného bajtu až do délky segmentu. Po bezchybném přijetí se generuje potvrzovací zpráva ACK. Pořadové číslo přijatého bajtu ve zprávě ACK vyjadřuje číslo následujícího bajtu, který je příjemce připraven přijmout. Příjemce tedy potvrzuje, že správně přijal vše až do pořadového čísla přijatého bajtu mínus jedna.

2.2.1 ZPŮSOBY POTVRZOVÁNÍ

Pro generování potvrzení u TCP se používá technika zpožděného odesílání ACK, tzv. piggybacking, který zajišťuje snížení objemu režie spojeného s odesíláním krátkých segmentů

u interaktivních aplikací, jako je například telnet nebo příkazový kanál FTP. Potvrzování přijatých dat neprobíhá okamžitě, ale se zpožděním (typicky 200 ms, maximálně 500ms podle RFC 1122). Aplikace klienta může používat i tzv. Nagleův algoritmus, kdy klient nečeká na vypršení časovače pro zpožděné odeslání, ale posílá data až v době, kdy dorazí data od protější strany. Nagleův algoritmus tedy pozdrží odpověď ze serveru pro klienta, pokud je na lince delší odezva. TCP garantuje taktéž zachování pořadí aplikačních dat. Segmenty přichází aplikační vrstvě příjemce ve stejném pořadí, v jakém byly vyslány.

2.2.2 ZPŮSOB NASTAVENÍ ČASOVAČE

Správné nastavení časovače je důležité, pokud je totiž časovač nastaven na krátký čas, může docházet ke zbytečnému opětovnému vysílání segmentů, z čehož vyplývá následovné plýtvání kapacity sítě. Naopak pokud je časovač nastaven na příliš dlouhou dobu, není využitá plná kapacita sítě a přijímací aplikace musí zbytečně dlouho čekat na data, která se ztratila přenosem v síti. Většina implementací protokolu TCP se snaží odhadnout aktuální dobu odezvy *RTT* na základě doby odezvy posledních potvrzených segmentů. Využívá se exponenciální průměrování, specifikované v RFC 739, které lze popsat vztahem:

$$SRTT = \alpha * SRTT + (1 - \alpha) * RTT. \tag{2.1}$$

kde *SRTT* je odhad *RTT*, α je vyhlazovací faktor
Konstanta α se pohybuje v intervalu $0 < \alpha < 1$, volí se obvykle 0.9, tím lze dosáhnout, aby odhad závisel z většího procenta na předcházející hodnotě *SRTT*, ale jen z malého procenta na nově naměřené hodnotě *RTT*. Takovouto volbou malé změny *RTT* při různých podmínkách v síti hodnotu *SRTT* příliš nezmění. Vlastní interval časovače *RTO* se podle původní specifikace protokolu TCP vypočítá jako:

$$RTO = \beta.SRTT. \tag{2.2}$$

kde β je predikovaná doba odezvy, doporučuje se nastavit $\beta = 2$ a *SRTT* je odhad *RTT*
Tento algoritmus byl poté ještě pozměněn a kromě sledování odhadu *SRTT* je potřeba sledovat i změnu *RTT*. Takto by se měl interval časovače více přizpůsobit změnám podmínek při přenosech. *RTO* se poté vypočítá celkově podle následujících měření:

- odhadem kolísání doby odezvy průměrnou absolutní odchylkou (*RTT* Variance Estimation)
- exponenciálním prodlužováním *RTO* opakovaného vysílání (*RTO* Exponentional Backoff)

Používá se také Karnův algoritmus (Karn's algorithm), který:

1) Nebere do úvahy doby odezvy *RTT* měřené pro segmenty, u kterých bylo nutné opakovat vysílání
2) V případě opakovaného vysílání hodnota časovače je stanovena využitím vztahu:

$$RTO^{(m+1)} = q * RTO^{(m)}.$$

3) U následně odeslaných segmentů využívá prodlouženou hodnotu *RTO* vypočítanou algoritmem backoff pro nepotvrzený segment a to tak dlouho, dokud nepřijme potvrzení k segmentu, který byl vyslán pouze jednou.

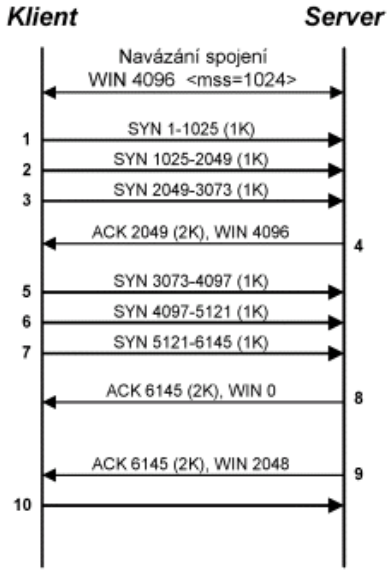
2.2.3 ZPŮSOBY PŘEPOSÍLÁNÍ

Při ztrátě segmentu nebo chybném přijetí TCP zajišťuje svými mechanizmy přeposílání chybějících segmentů. Může taky nastat situace, kdy se přijímač nemůže přesvědčit, že potvrzení ACK dorazilo k odesílateli, tím může docházet k přeposílaní segmentů nejen kvůli ztrátě vyslaných dat. Způsob, jakým je řešena spolehlivost výrazně ovlivní přenosové vlastnosti. Na straně vysílače se po vyslání segmentu spouští časovač *RTO*, a pokud nepřijde do uplynutí doby časovače potvrzení o správném přijetí, vysílač znovu přepošle segment.

Pokud dojde ke ztrátě segmentu, přeposílání se řídí podle pravidel, která se dohodnou na počátku spojení (standardně kumulativní potvrzovací systém, novější varianty podporují selektivní potvrzovací systém SACK (RFC 2018). Kromě vypršení časovače se ke ztrátě segmentu využívá také signál přijetí tří duplicitních potvrzovacích ACK – dupACK, který se nazývá algoritmus rychlého přeposlání. Pokud příjemce obdrží data v jiném pořadí, než ve kterém byly vyslané, mohlo dojít pouze k situaci, že v síti došlo k přeuspořádání dat, ale eventuálně brzy dorazí. Může se taky jednat o signál, že v sítí se ztratily některé segmenty, ale další segmenty stále přicházejí k příjemci. Algoritmus rychlého přeposlání umožňuje zopakovat pouze ztracený segment ještě před vypršením časovače a nikoliv nutnost opakování všech nepotvrzených segmentů po vypršení časovače.

2.3 ŘÍZENÍ PŘENOSOVÉ RYCHLOSTI TCP

Jak již bylo krátce popsáno v úvodu, TCP provádí kontrolu toku dat mezi vysílačem a přijímačem. Rychlost příchodů potvrzení ACK k vysílači je ovlivněna úzkým místem mezi vysílačem a přijímačem, kterým může být buďto přijímací uzel nebo přenosová síť. Důvodem zahlcení přijímače může být například příliš velké vytížení procesoru nebo velké množství segmentů přicházejících přes další TCP spojení. Aby odesílatel nezahlcoval vyrovnávací paměť přijímače a kvůli tomu nedocházelo ke ztrátám segmentů, omezuje odesílatel množství odeslaných a dosud nepotvrzených dat. Kontrola toku je realizována systémem posuvného okénka. Okénko *owin* je množství dat, které mohou být v dané chvíli vyslané po síti, aniž by měl jejich příjem potvrzen. Okénko *owin* závisí na velikosti vyrovnávací paměti přijímače, okénka *rwnd*. Platí, že *owin* <= *rwnd*. Při každém přenosu si obě strany posílají velikosti svých oken. Pokud má přijímač zaplněné vyrovnávací paměti, může nastavit *rwnd* na nulu a tím úplně zastavit vysílání. Tento jednoduchý mechanismus byl součástí TCP dříve než předcházení zahlcení. Vysílající entita TCP neví, jestli rychlost přicházejících potvrzení odpovídá stavu sítě (řízení propustnosti) nebo stavu koncového uzlu (řízení toku dat). Obrázek popisuje navázání spojení mezi vysílací a přijímací entitou TCP, dohodnutí se na velikosti okna *owin*, dohodnutí se na velikosti *MSS* a systém posuvného okénka.



Obr. 2.3 - Výměna zpráv mezi vysílací a přijímací entitou TCP

2.4 ŘÍZENÍ PROPUSTNOSTI, MECHANISMY PŘEDCHÁZENÍ ZAHLCENÍ

Aktuální rychlost přenosu TCP spojení R je dána vztahem:

$$R = owin / RTT \ [B/s]. \tag{2.3}$$

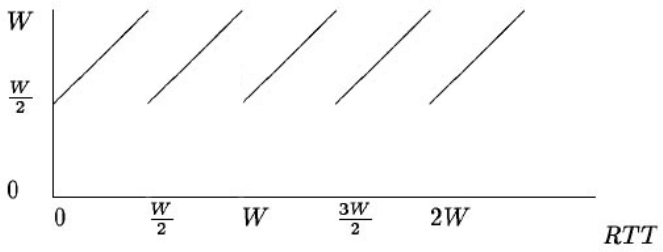
kde *owin* je velikost okna na straně vysílače v bajtech, a *RTT* je čas od vyslání paketu k příjmu potvrzení v sekundách.
Tento předpoklad platí pouze na lince s konstantním *RTT*, kdy má linka dostatek propustnosti, aby se ve směrovačích neplnily výstupní fronty. Platí, že velikost okna může mít největší hodnotu:

$$owin = \min (cwnd, rwnd) \ [B]. \tag{2.4}$$

Mechanizmus posuvného okna *owin* je určen především pro řízení provozu mezi koncovými uzly, prakticky však nastavení velikosti okna musí zohledňovat i potřeby řízení propustnosti. Maximální velikost okna je určená 16 bitovým polem v TCP hlavičce, schopné adresovat 2^{16}-1, tj. 65535 B, což by mělo být dostačující pro většinu aplikací. Může dojít ovšem i k situaci, kdy velikost okna není dostačující. Řešením je volba TCP Window Scale, která se může poslat v hlavičce úvodního SYN segmentu ve volitelném poli options. S její pomocí se zvětšují jednotky pro velikost klouzavého okna, tedy že se například namísto bajtů interpretuje v kilobajtech.

2.4.1 ŘÍZENÍ PROPUSTNOSTI

Vysílač podle různých typů signálů o blížícím se či již nastalém zahlcení sítě vypočítává limit pro velikost okénka zahlcení *cwnd*. Musí platit *owin*<=*cwnd*. Okénko zahlcení *cwnd* se zvyšuje aditivně o konstantu za každou hodnotu doby *RTT* a při signálu zahlcení se multiplikativně zmenší (AIMD). Jakmile se postupně zvyšuje okénko *cwnd*, hranice, za kterou je větší pravděpodobnost zahlcení sítě, se označuje *SSTHRESH*. *SSTHRESH* se udržuje v násobcích *MSS*. Okénko zahlcení má typický pilovitý průběh, viz obr.4



Obr. 2.4 - Pilovitý průběh závislosti okénka zahlcení *cwnd* v [*B*] na době *RTT* v [*s*]

2.4.2 MECHANISMY PŘEDCHÁZENÍ ZAHLCENÍ

První specifikace TCP RFC 793 zahrnovala pouze kontrolu toku pro ochranu před zahlcením vyrovnávací paměti přijímače, nespecifikovala mechanizmy na dynamické přizpůsobení rychlosti vysílání jako reakci na zahlcení. Zahlcení sítě je stav, kdy do sítě vstupuje víc dat, než je síť v daném okamžiku schopná přenést. Může se jednat o zahlcení vnitřních vyrovnávacích front směrovače, pokud se rychlá vstupní linka snaží dostat data na pomalou výstupní linku nebo pokud více vstupních linek směřuje data na stejný port jedné výstupní linky. Taky může dojít k situaci, kdy směrovač nezvládá zátěž, která skrze něj prochází, i když výstupní fronty jsou nezahlcené. Protože fronty směrovače nemohou mít nekonečné délky (a tím také nekonečné zpoždění), dochází k zahazování paketů. To má velký dopad na přenosové charakteristiky protokolu TCP. Signál zahlcení je daný ztrátou paketu, ovšem ne každá ztráta paketu musí znamenat zahlcení. Například jedna z variant TCP – SACK označí signál zahlcení až ztrátu více paketů během jednoho *RTT*.

Algoritmy na kontrolu zahlcení byly navrhnuté už v r. 1986. Tyto mechanizmy pracují na koncových stanicích spojení a způsobují, že TCP zpomalí vysílání v době zahlcení sítě. Existuje několik řešení pro efektivní řízení velikosti posuvného okénka vysílače. Jedná se o algoritmy specifikované v RFC 2001 a RFC 2581, algoritmy pro předcházení zahlcení, zajištění vysokého využití kapacity linky a férovosti ve sdílení linky s ostatními toky:

- Pomalý start
- Vyhýbání se zahlcení
- Rychlé přeposílání
- Rychlé zotavení
- Omezené vysílání

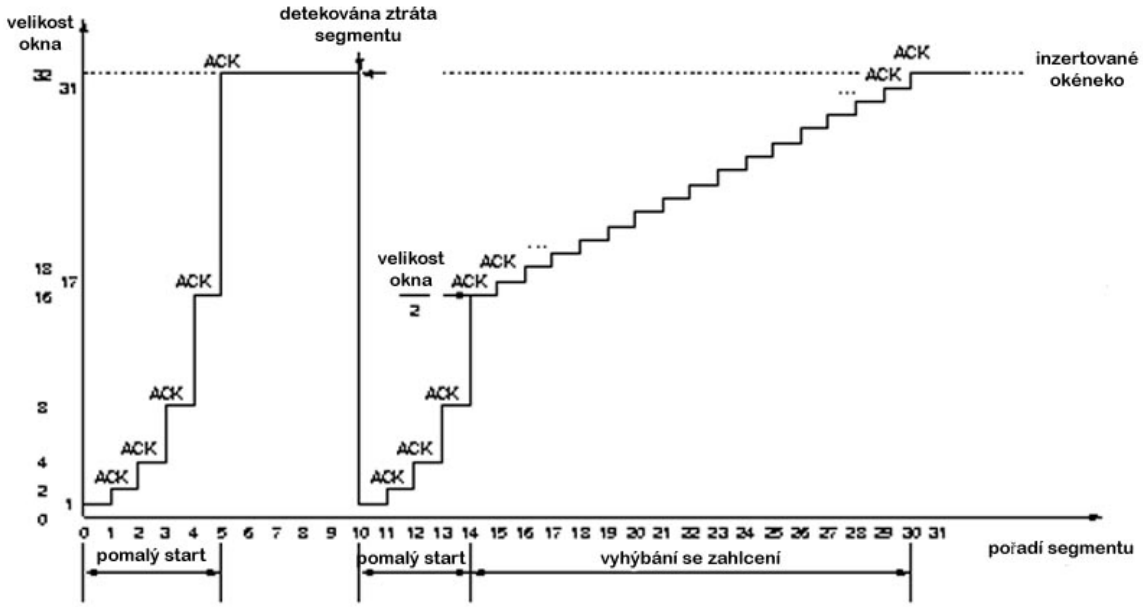
Pomalý start

Po navázání spojení s protější stranou TCP nezačne ihned vysílat plnou rychlostí kapacity výstupní linky až do naplnění velikosti okna *rwnd* na straně příjemce. Nejdříve odešle jeden segment, jakmile dorazí potvrzení ACK, posílá dva segmenty, poté čtyři atd. Rychlost vysílání se exponenciálně zvyšuje až na hodnotu *SSTHRESH*, kdy už je větší pravděpodobnost ztráty segmentů.

Vyhýbání se zahlcení

Jakmile je hodnota *cwnd* větší než SSTHRESH, pak odesílání dvojnásobku by již mohlo způsobit zahlcení. Zahlcení je detekováno ztrátou segmentu. Algoritmus se snaží vyhnout zahlcení tím způsobem, že změní nárůst *cwnd* z exponenciální funkce na lineární. Pokud dojde ke ztrátě, sníží aktuální rychlost na polovinu a změnou tempa zvětšování rychlosti vysíláni se pomalu blíží k hranici zahlcení. Algoritmus vyhýbání se zahlcení spolupracuje s algoritmem pomalého startu, a je popsán následovně:

1) Při inicializaci spojení se nastaví *cwnd* = 1 segment a *ssthresh* = 65535 B (odpovídající počet segmentů)
2) Při detekci zahlcení nastavení *ssthresh* = *cwnd*/2.
3) Uvedení *cwnd* do počáteční hodnoty, tj. *cwnd* = 1 segment a zahájení fáze pomalého startu, dokud *cwnd* nedosáhne hodnotu *cwnd* = *sstrhes*.
4) Od *cwnd* ≥ *ssthres* velikost *cwnd* bude po každém uplynutí doby odezvy zvýšena o jedno.



Obr. 2.5 - Spolupráce algoritmů rychlého startu a vyhýbání se zahlcení

Pomalý start a předcházení zahlcení nutí TCP protokol redukovat hodnotu *cwnd* na jedna pokaždé, když je detekována ztráta segmentu. Pokud zahlcení trvá delší čas, množství dat poslané do sítě klesá exponenciálně. Tímto se zajistí vyprázdnění front ve směrovačích a odstranění zahlcení sítě. Při návrhu těchto algoritmů se předpokládá, že ztráta paketů je způsobena při přetížení a jen malé procento způsobuje ztráta paketů na lince

Rychlé přeposílání

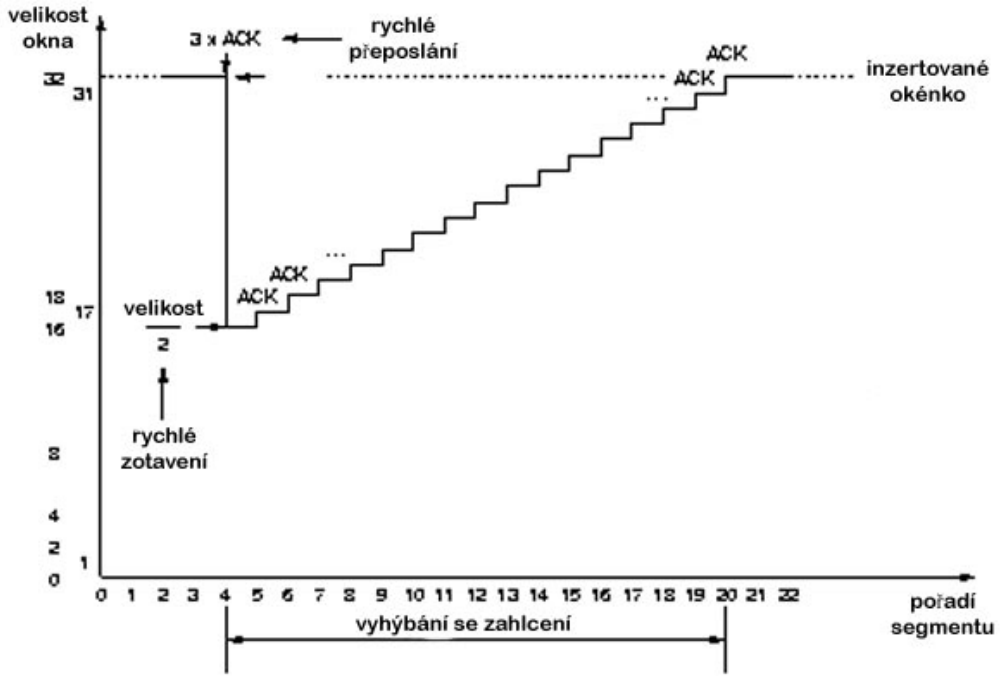
Původní specifikace TCP předpokládá při příjmu zprávy dupACK ztrátu segmentu a znovu ho ihned přeposílá. Algoritmy rychlého přeposlání a rychlého zotavení zvyšují propustnost protokolu při ztrátě malého počtu segmentů nebo přeuspořádání pořadí vysílaných segmentů. V této situaci mohou zlepšit výkonnost určenou hodnotou *RTO*. Pokud dojde ke ztrátě segmentu, příjemce to zjistí příchodem dalších segmentů mimo pořadí. Příjemce je povinen po přijetí segmentu mimo pořadí zopakovat (duplikovat) potvrzení posledního správně přijatého segmentu. Protože se TCP segmenty balí do IP datagramu a každý IP datagram může přes síť putovat jinou cestou, může dorazit k příjemci později než následující segment. V této době již však příjemce odeslal duplikované potvrzení. Pokud ztracený segment stále nepřichází, příjemce po vyslání třetího dupACK oznamuje odesílateli, že je potřeba znovu vyslat ztracený segment. Jakmile příjemce obdrží ztracený segment (příjemce nezahodí žádný z následujících segmentů mimo pořadí), potvrdí přijetí všech přijatých segmentů. Algoritmus

rychlého přeposílání tedy umožňuje zopakovat pouze ztracený segment a ne nutnost opakování všech nepotvrzených segmentů ještě dříve, než vyprší časovač na straně odesílatele.

Rychlé zotavení

Jakmile vysílač obdrží zprávu dupAck, znamená to, že data jsou sítí stále vysílány, protože TCP generuje tuto zprávu jen tedy, pokud přicházejí další segmenty (v jiném pořadí než byly vyslány). Původní specifikace by nastavila *ssthresh* = *cwnd*/2 a hodnota *cwnd* by se nastavila na 1. Zahájil by se proces pomalého startu s exponenciálním náběhem až do hodnoty *ssthresh* a dále by pak pokračoval už lineární nárůst. Myšlenka je taková, abychom ochránili linku před úplným vyprázdněním po fázi rychlého přeposlání. Algoritmus rychlého zotavení zabraňuje TCP spojení, aby se rychlost vysílání po vykonání rychlého přeposlání snížila na nulu (snížení *cwnd* na nulu se tedy nyní provádí jen při vypršení časovače). Algoritmus rychlého zotavení při příjmu třetího dupACK nastaví cwnd na polovinu, zopakuje odeslání TCP segmentu a pokračuje dál s lineárním nárůstem. Výsledkem je tedy úplné vynechání počátečního exponenciálního náběhu. Algoritmus je popsán následovně:

1. Pokud příjemce obdrží třetí duplikát, pak:
    a) nastaví *SSHTRESH* na *CWND*/2,
    b) zopakuje odeslání TCP segmentu,
    c) nastaví *CWND* na *SSHTRESH*+3**MSS*
    (Důvodem zvýšení hodnoty *cwnd* o 3**MSS* v posledním kroku je doručení tří dalších segmentů po ztraceném segmentu indikovaným příchodem tří duplikátů potvrzení)
2. Po přijetí čtvrtého a dalšího duplikátu odesilatel vždy pokaždé zvětší *CWND* o velikost segmentu (*MSS*).
3. V případě, že ztracený segment je konečně potvrzen příjemcem, pak odesilatel nastaví *CWND* na hodnostu *SSTHRESH*.



Obr. 2.6 - Spolupráce algoritmů rychlého zotavení a rychlého přeposlání

Omezené vysílání

Algoritmus omezeného vyslání se snaží vyvolat odeslání nového segmentu v případě, že:

a) Byly přijaty dva duplikáty potvrzení, tj. celkem tři potvrzení pro stejný segment.
b) Okno vysílače nastavené od přijímače umožňuje vyslání segmentu.
c) Objem nepotvrzených dat po odeslání nového segmentu nepřesáhne hodnotu *cwnd* +2, tj. vysílač může odeslat maximálně dva segmenty nad rámec velikosti okna zahlcení.

Mechanizmus omezeného vyslání umožňuje vyslat omezený objem dat i přes omezení, vyplívající z mechanizmů ochrany proti zahlcení.

3 VÝVOJ TCP

Systémy kontroly zahlcení byly postupně nasazované do různých implementací TCP a poté byly postupně optimalizované

TCP Tahoe

Implementovaný v 4.3BSD v roce 1988, obsahoval první tři mechanizmy kontroly zahlcení- pomalý start, předcházení zahlcení, a rychlé přeposlání. Ztráta je detekována pokud vyprší

časovač, předtím než je doručen ACK. Tahoe poté zmenší velikost okénka na 1 *MSS* a přejde do fáze pomalého startu.

TCP Reno

Implementovaný v 4.3BSD v roce 1990, obsahoval čtyři mechanizmy kontroly zahlcení- pomalý start, předcházení zahlcení, rychlé přeposlání, rychlé zotavení. Ztráta je detekována pokud jsou doručeny 3 duplikátní ACK (3 dupACK potvrzující stejný segment, nemění velikost okénka). Reno zmenší velikost okénka na polovinu, přejde do fáze rychlého přeposlání a poté vstoupí do fáze rychlého zotavení. Pokud vyprší časovač, použije se pomalý start stejně jako u Tahoe. Při Fast Recovery (jen u TCP Reno) TCP přeposílá chybějící segment, který byl signalizovaný přijetím 3 dupACK a čeká na potvrzení celého okna předtím, než přejde do fáze vyhýbání se zahlcení. Pokud pak nepřijde žádné potvrzení, TCP Reno vyprší časovač a vstoupí do fáze pomalého startu. Reno poté zmenší velikost okénka na 1 *MSS* stejně jako Tahoe
Problémy TCP RENO: Reno funguje velmi dobře v sítí, kde dochází pouze k malé ztrátě paketů. Ovšem pokud dochází k vícenásobné ztrátě paketů jednoho okna, poté nefunguje RENO příliš dobře a výkon při takové ztrátě je stejný jako TCP TAHOE.

TCP Vegas

Diskutovaný v r. 1994, nový pohled na výskyt zahlcení, Sleduje RTT svých paketů a při zvyšování této hodnoty zmenšuje hodnotu okénka zahlcení *cwnd* a naopak. V oblasti férovosti při soupeření o linku se staršími implementacemi TCP je v nevýhodě, protože při algoritmu pomalého startu se rychlost se zvyšuje o polovinu pomaleji

TCP SACK

Definovaný v RFC 2180 v r.1996 – vylepšuje propustnost RENO v případech, kdy dojde ke ztrátě více paketů, při ztrátách segmentů posílá příjemce vysílající straně ACK s rozšířením o SACK, který oznamuje, které pakety přijal a které mu chybějí, také vylepšuje problém New-Reno, tedy přeposlání více než jednoho paketu během RTT. Vysílací strana poté může přeposlat pouze chybějící pakety, čímž velmi šetří zašuměnou linku (např. Wifi).
SACK TCP vyžaduje, že segmenty nebudou potvrzovány kumulativně, ale selektivně. To

znamená, že každý ACK má pole, které popisuje, které segmenty jsou potvrzovány. Tímto dostává vysílač obrázek, které segmenty už byly potvrzeny, a které jsou ještě v síti.

TCP New Reno

Specifikovaný v RFC 2582 v r. 1999 – vylepšuje propustnost TCP Reno, pokud dojde ke ztrátě více paketů, které patří do údajů v jednom okně. Zdokonaluje opakované vysílání TCP Reno během fáze rychlého zotavení. Stejně jako Reno po přijetí 3 dupACK vstoupí o fáze rychlého přeposlání, ovšem neopustí fázi rychlého zotavení, dokud nejsou potvrzeny všechny data, která byla poslána během fáze rychlého zotavení. Tímto se tedy předejde nedostatku Reno, který vícekrát redukuje velikost CWND.
Problémy: New Reno trpí nedostatkem, že zabere dobu RTT, než detekuje ztrátu každého paketu. Jakmile je přijatý ACK pro první přeposlaný segment, pouze poté můžeme odvodit, které další segmenty byly ztraceny

4 NASAZENÍ PROTOKOLU TCP VE VYSOKORYCHLOSTNÍCH A BEZDRÁTOVÝCH SÍTÍCH

Mechanizmy kontroly zahlcení v protokolu TCP vyřešily problém zahlcení a kolapsu sítě, ovšem prokázalo se, že se příliš nehodí do vysokorychlostních sítí (řádově Gigabity/s) a také do populárních bezdrátových sítí. Ve vysokorychlostních sítích TCP sonduje šířku pásma příliš pomalu. Lineární zrychlování růstu velikosti okénka je příliš pomalé a výsledkem předcházení zahlcení je nevyužití plné kapacity linky a výrazné poddimenzování možností přenosového pásma linky.
V bezdrátových sítích neplatí předpoklad, na kterém je postavena kontrola zahlcení TCP. V těchto sítích dochází ke ztrátám ne kvůli zahlcení sítě, ale kvůli chybovosti na lince. Jakmile TCP zjistí ztrátu paketů, tak sníží rapidně rychlost vysílání. Kvůli tomu se při krátkých a náhodných interferencích snižuje přenosová rychlost TCP protokolu a dochází ke slabému využití skutečné přenosové šířky pásma.

4.1 CHARAKTERISTIKA VYSOKORYCHLOSTNÍCH SÍTÍ

4.1.1 BANDWIDTH-DELAY PRODUCT, OMEZENÁ VELIKOST OKÉNKA

TCP opravuje chybějící nebo porušená data jejich přeposíláním z vyrovnávací paměti odesílatele, aby mohly být připravena ve správném pořadí ve vyrovnávací paměti přijímače a následně odeslána aplikaci. Takovýto proces vyžaduje, aby se celé okénko vešlo jak do vyrovnávací paměti odesílatele, tak i přijímače. Takové vyrovnávací paměti mají standardní velikosti, které mohou být pozměněné jen aplikací použitím systémového volání knihovny. Navíc operační systém může mít limity, které omezují maximální vyrovnávací paměť, kterou může aplikace požadovat. Ideálně by měla být propustnost TCP omezena pouze některým skutečným úzkým hrdlem, jako je rychlost disku nebo kapacita sítě. Nicméně v praxi je propustnost často limitována velikostí vyrovnávacích pamětí soketů, nebo malým okénkem zahlcení v souvislosti s určitým problémem v síti. V RFC 1323 se uvádí, že protokol TCP je navrhnutý tak, aby se adaptoval na přenosové charakteristiky linky s přenosovou kapacitou 100 b/s – 10 Mb/s a zpožděním na lince 1 ms – 100 s.

Množství dat, které mohou být vyslané do sítě, jsou označené termínem „Bandwidth Delay Product“, zkráceně *BDP*

$$BDP = BW * RTT \tag{4.1}$$

kde *BW* je šířka pásma v bitech/s, a *RTT* je doba v sekundách, doba od vyslání a přijetí ACK Vyrovnávací paměti vysílače i přijímače by měly být dostatečně velké, aby vysílač mohl naplnit přenosovou kapacitu linky. Předpokládejme, že úzké hrdlo spojení má přenosovou kapacitu *BW* b/s a linka mezi vysílačem a přijímačem má *RTT* sekund. Pokud není na lince žádný jiný soupeřící přenos, TCP bude schopný naplnit linku, pokud nastaví okno velikosti *BW*RTT*, tedy *BDP*. Kontrola toku TCP vyžaduje, aby okénka vysílače i přijímače byla stejně velká. Pokud je velikost vyrovnávací paměti vysílače nebo přijímače menší než *BW*RTT*, přenosová kapacita nebude dostatečně využitá.

Tím jak se v posledních letech zvedala kapacita propustnosti sítí, operační systémy postupně zvyšovaly standardní velikosti vyrovnávacích pamětí z běžných hodnot 8 kB na 64 kB. I přesto je to stále příliš málo pro dnešní vysokorychlostní sítě. Pokud máme například linku 10 Gbps, která má 200ms *RTT*, vyrovnávací paměť TCP by měla být přibližně (10 Gbps /

8bit/byte * 0.2 sekund) = 238 MB. Takovéto velikosti jsou ovšem nadměrné v reálném prostředí.

OS	Maximální velikost vyrovnávací paměti	Maximální propustnost
Windows NT	64KB	5Mbps
Linux 2.4	128KB	10Mbps
Free BSD	262KB	20Mbps
Solaris	1000KB	80Mbps

Tab. 4.1 - Velikosti vyrovnávacích pamětí některých OS a maximální možná propustnost při *RTT*=100ms

V souvislosti s vyrovnávací pamětí má tedy zmenšení propustnosti následující důvody:

a) Velikost okénka owin je v hlavičce TCP omezena na 16 b, což odpovídá maximu 65535B
b) Některé operační systémy alokují pro každé TCP spojení paměť, která může být ještě menší než 64kB

Důvod a) se dá vyřešit zapnutím funkce window scaling, při které se násobí okénko na obou stranách konstantou dohodnutou při otevírání spojení. Například pro plné využití volné kapacity v lokální síti s *BW* = 100Mb/s a *RTT* = 1ms vychází potřebná velikost okénka 12 kB. Pro využití volné kapacity v síti s *BW* = 10Gb/s a *RTT* = 250ms vychází potřební velikost okénka 298MB. Nejlepší řešení by byla automatická konfigurace velikosti okénka na straně vysílače i přijímače. Na toto téma vzniklo několik prací, například [1, 2].

Důvod b) se dá odstranit vyladěním parametrů operačního systému. Takovýto nástroj je pro WINDOWS například program SG TCP Optimizer [3], který dokáže například spočítat *BDP* pro nalezení nejlepšího okénka pro specifickou rychlost připojení. Dokáže také nastavovat další TCP parametry jako *MTU*, velikost okénka *rwin*, zapnutí selektivního potvrzování SACK nebo volbu počtu maximálním dupACK pro následné přeposlání.

4.1.2 POMALÝ NÁVRAT K OPTIMÁLNÍ PROPUSTNOSTI

Dalším problémem v souvislosti s použitím TCP ve vysokorychlostních sítích je pomalý návrat k optimální propustnosti při přenášení objemných dat. Při každé menší ztrátovosti, TCP vyhodnotí situaci jako zahlcení sítě a rekcí je nejen opakovaný přenos ztraceného segmentu, ale taky velké zpomalení vysílání. Pokud by se rychle situace napravila a dosáhlo se během krátké doby optimální propustnosti, byla by situace v pořádku. TCP ale velice pomalu zvyšuje velikost okna (po bytech). V tabulce č. 3 podle [4] jsou uvedené časy zotavení se protokolu TCP při ztrátách paketů a při různých přenosových rychlostech. Tabulka zobrazuje taky jednu z modifikací algoritmu pro předcházení zahlcení v TCP pro vysokorychlostní sítě – Scalable TCP. Předpokládá se konstantní *RTT*=200ms. Na počátku simulace se předpokládá, že TCP vysílá na plné rychlosti linky. Poté dojde ke ztrátě paketů, na který reaguje TCP snížením rychlosti vysílání na polovinu. V tabulce jsou uvedeny časy, za jak dlouho se protokol TCP dostane zpět na plnou rychlost linky.

Přenosová rychlost	Standardní TCP - čas zotavení	Scalable TCP – čas zotavení
1Mbps	1.7s	2.7s
10Mbps	17s	2.7s
100Mbps	2min	2.7s
1Gps	28min	2.7s
10Gps	4hod43min	2.7s

Tab. 4.2 - Časy zotavení na plnou přenosovou rychlost při ztrátě paketu

Pro použití TCP v sítích s vysokým Bandwidth-delay product existují tedy dva závažné problémy:

1) TCP se nemůže přiblížit k plnému využití přenosové kapacity sítě.
2) Jsou potřeba velké vyrovnávací paměti

4.2 TCP PRO SÍTĚ S VYSOKÝM BANDWIDTH-DELAY PRODUCT

S příchodem aplikací pro nové generace, které vyžadují ustálené přenosové rychlosti v řádu gigabitů za sekundu, a které musí být schopné spolehlivě přenášet velké množství dat v krátkém čase na velké vzdálenosti, široce používaný TCP protokol se stává úzkým hrdlem přenosu. Existují dva základní přístupy ke zrychlení TCP. První je paralelizace existujícího TCP spojení a druhá je zasáhnout do algoritmu TCP pro zrychlení jednotlivého TCP přenosu. Používání paralelních TCP proudů jako prostředku pro zvýšení výkonu TCP je přístup, který již kdysi existoval. Například původní specifikace HTTP dovolila používání paralelních TCP spojení pro stahování jednotlivých částí www stránek (ačkoliv HTTP 1.1 se vrátil k sekvenčnímu stahování, protože v tomto případě režijní náklady spojení při navazování spojení převyšovaly výhody paralelních TCP spojení). Aby ovšem paralelní TCP spojení fungovalo správně, potřebujeme již mít shromážděné všechny data (nebo alespoň musíme vědět, kde jsou všechny části dat umístěné). Tam kde jsou data generované v reálném čase (jako jsou například observatoře hvězdáren nebo pozorování srážek částic) ve velkých množstvích, neexistuje jiná volba, jak nakládat se skupinou dat při odesílání, než jako se sériovým přenosem s využitím vysokorychlostního TCP protokolu.
Mnoho výzkumných týmů navrhlo zdokonalení a přizpůsobení TCP pro vysokorychlostní sítě, jakými se jeví nejvíce perspektivní High speed TCP [5], Scalable TCP[6] a HTCP [7]

Highspeed TCP (HSTCP)

High-speed TCP přizpůsobuje kontrolu zahlcení pro TCP spojení s velkými okénky. HSTCP je navrhnutý k tomu, aby vylepšil důsledky bitových chybovostí na dlouhotrvajících TCP přenosech přes BDP sítě. Chování současného standardního protokolu TCP omezuje okénko zahlcení, které může být dosažené v reálných prostředích. Například při standardním TCP spojení s pakety o velikosti 1500 bytů a RTT = 100ms, pro dosáhnutí propustnosti 10Gb/s by bylo zapotřebí okénko zahlcení o průměrné velikosti 83333 paketů a poměr ztrátovosti nejvýše jedna ztráta při zahlcení každých 5000.000.000 paketů, což je velmi nerealistické omezení. High-speed TCP je navržený aby měl jinou odezvu v prostředích s velmi malým zahlcením. V sítích s malou ztrátou paketů, nanejvýš 10^{-3} se nechová jako standardní TCP. Díky tomu, že HSTCP zanechává nezměněné chování TCP v prostředích se středním až silným zahlcením, nezvyšuje tak risk kolapsu sítě při přetížení. V prostředích s velmi nízkým poměrem ztrátovosti HSTCP představuje mnohem agresivnější funkci odezvy. Funkci odezvy

High-speed TCP určují 3 parametry: *low_window*, *high window* a *high_p*; parametr *low_window* se používá k ustanovení bodu přechodu. Pokud aktuální okénko zahlcení odpovídá aspoň parametru *low_window*, funkce odezvy HSTCP používá stejnou funkci odezvy jako běžný TCP, pokud je aktuální okénko zahlcení vetší než *low_window*, používá funkci odezvy pro HSTCP. Parametry *high_window* a *high_p* se používají pro specifikaci vrchního limitu funkce odezvy pro HSTCP.
Funkce odezvy HSTCP je reprezentována novými parametry aditivního zvětšení a násobného snížení. Při fázi vyhýbání se zahlcení se výpočet okénka zahlcení provádí následujícím způsobem:

- při příchodu ACK: $cwnd = cwnd + a*(cwnd)/cwnd$
- při ztrátě paketu: $cwnd = cwnd - b(cwnd) * cwnd$

HighSpeed TCP tedy zvyšuje okénko zahlcení 1 segment pro přenosové rychlosti do 10Mb/s, 6 segmentů pro přenosové rychlosti do 100Mb/s, 26 segmentů pro přenosové rychlosti do 1Gb/s a 70 segmentů pro přenosové rychlosti do 10Gb/s. Jinými slovy, čím větší rychlost TCP byla dosažena, tím větší akcelerace. Jako odpověď na ztráty paketů reaguje menším násobkem snížení, a to tak že pro 10Mb/s bude násobitel 1/2, pro 100Mb/s bude 1/3, pro 1Gb/s bude 1/5 a pro 10Gb/s bude 1/10.
HSTCP je tedy vhodný pro aplikace s objemnými přenosy, protože je schopný udržovat vysokou výkonnost při různých stavech sítě. Přizpůsobení parametrů je navrženo k tomu, aby mírně sondoval síťovou kapacitu během počáteční fáze pomalého startu. Pokud dojde ke ztrátě paketů, po fázi pomalého startu HSTCP uvolní podstatně méně šířky pásma sítě než 1/2 *cwnd*, a obnovuje rychlost mnohem agresivněji než standardní TCP.

Scalable TCP (STCP)

Scalable TCP se pokouší odstranit vztah mezi správou TCP okénka a časovým intervalem RTT. Ve fázi vyhýbání se zahlcení, tradiční TCP při odpovědi na každý ACK zvětší okénko zahlcení cwnd vysílače velikostí 1/cwnd tím, že okénko je postupně zvětšováno jedním segmentem každý interval RTT. Podobně půlení okénka při ztrátě paketu může být vyjádřeno jako snížení velikosti okénka na cwnd/2. STCP Zlepšuje výkon TCP použitím pevné hodnoty aditivního zvětšení *a* a násobku snížení *b*, který je menší než 1/2 cwnd používaný ve standardním algoritmu pro předcházení zahlcení. Takovéto adaptivní faktory dělají STCP

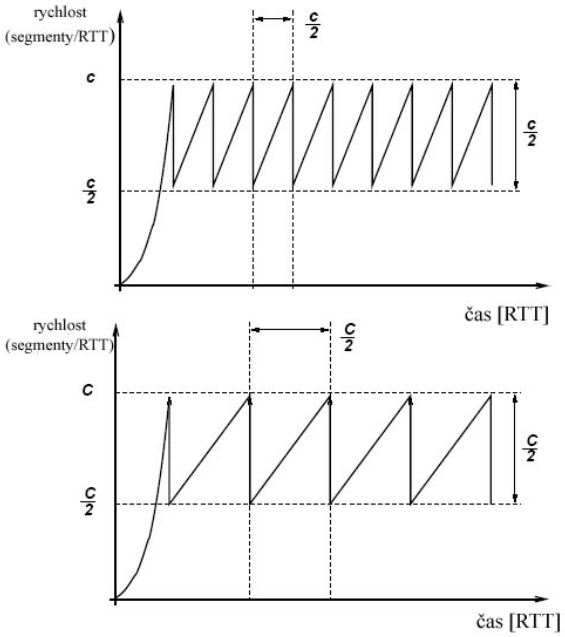
mnohem agresivnější, něž standardní TCP díky obsazení větší šířky pásma při sondování síťové kapacity a uvolnění menší šířky pásma při ztrátě paketů. Scalable TCP mění algoritmus pro aktualizaci okénka přetížení následujícím způsobem:

- při příchodu ACK do doby *RTT*, kdy nebylo detekováno zahlcení:
  $cwnd = cwnd + a$, kde a je konstanta v intervalu $0 < a < 1$
- při první detekci zahlcení během dané doby RTT: $cwnd = cwnd - [b*cwnd]$, kde b je konstanta v intervalu $0 < a < 1$

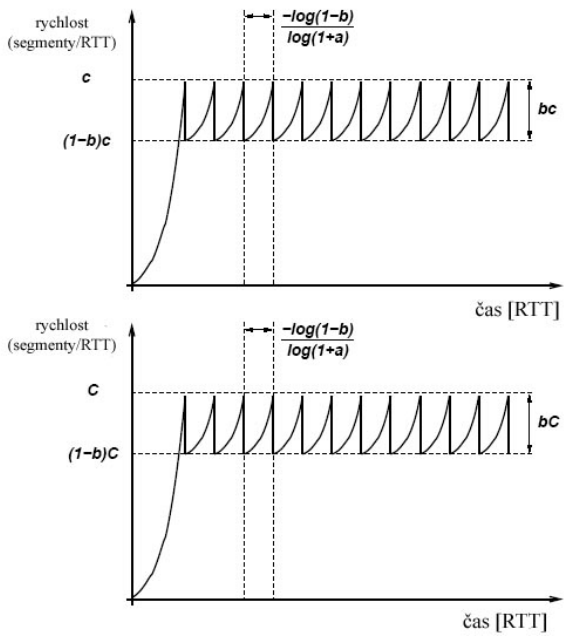
Konstanty se volí a=0.01 a b=0.125 kvůli dopadu na provoz, vlastnosti alokace šířky pásma, rozdílné přenosové rychlosti a stabilitu.

Základní charakteristikou Scalable TCP je tedy použití násobného zvětšení okénka zahlcení, spíše nežli lineárního aditivního zvětšování. Tímto se dosahuje vyšší kmitočet sondování šířky pásma, častější a účinnější než u HighSpeed TCP. Frekvence sondování Scalable TCP není závislá na intervalu RTT, může být vyjádřena funkcí $f = log(1 - b) / log(1 + a)$. Díky tomu ve sdíleném úzkém hrdle v síti delší přenosové linky vykazují podobné chování jako kratší přenosové linky.

Obrázek 4.1 zobrazuje změnu okénka zahlcení TCP spojení při použití standardního TCP a obrázek 4.2 Scalable TCP přes vyhrazenou linku kapacity c nebo C, kde $c < C$. Doba zotavení při ztrátě segmentu standardního TCP je úměrná velikosti okénka a RTT. Doba zotavení při ztrátě segmentu u Scalable TCP je úměrná pouze *RTT*. Takováto pevná hodnota pro přenos dovolí Scalable TCP překonat tradiční TCP ve vysokorychlostních sítích.



Obr. 4.1 - Změna okénka zahlcení TCP



Obr. 4.2 - Změna okénka zahlcení STCP

HTCP

HTCP dosahuje lepších výsledku ve vysokorychlostních sítích zmenšením časového intervalu mezi událostmi přetížení. Signál, že TCP už navýšil dostupnou šířku pásma, je brán jako událost zahlcení a s rostoucí frekvencí těchto událostí TCP bude sledovat tuto metriku s větší přesností. Tvůrci HTCP pro dosažení takového sledování navrhují změnu funkcí zvětšování a snižování okénka zahlcení, otázkou je jakým způsobem. Kritickým faktorem je totiž úroveň citlivosti na ostatní konkurenční síťové přenosy a schopnost přiblížit se ke stabilizaci alokace dostupných síťových prostředků soutěžících konkurenčních přenosů. V rámci férovosti by přenosy s velkými okénky zahlcení měly redukovat jejich velikosti více, než přenosy s malými okénky zahlcení. To zaručí pohotové ustanovení dynamické rovnováhy mezi navázanými TCP přenosy a novými, které vstupují na stejnou linku.

HTCP využívá namísto cwnd uplynulý čas Δ od doby poslední události přetížení, což signalizuje BDP linku. Algoritmus AIMD se nyní mění jako funkce Δ. Parametr aditivního zvyšování AIMD je opatřen měřítkem podle RTT linky aby zvýšil férovost mezi soutěžícími přenosy s rozdílným časy RTT. Parametr násobného snížení AIMD je nastaven tak, aby využil kapacitu linky, je založen na odhadu zásobení pamětí fronty na lince.

- při příchodu ACK: $cwnd = cwnd + 2*(1-\beta)*f_\alpha(\Delta) / cwnd$
- při ztrátě paketu: $cwnd = g_\beta(B) * cwnd$

$$f_\alpha(\Delta) = \begin{cases} 1 & \Delta \le \Delta L \\ \max(\overline{f}\alpha(\Delta) T\min, 1) & \Delta \rhd \Delta L \end{cases}$$

$$g_\beta(B) = \begin{cases} 0.5 & \left|B(k+1) - B(k) / B(k)\right| \rhd \Delta B \\ \min(T\min / T\max, 0.8) & jinak \end{cases}$$

- kde Δ_L je specifikovaný práh, který používá standardní TCP algoritmus při $\Delta \le \Delta_L$
- kvadratická funkce navyšování $\overline{f}\alpha$ je navrhnutá jako $\overline{f}\alpha(\Delta) = 1 + 10(\Delta - \Delta_L) + 0.25(\Delta - \Delta_L)^2$
- T_{min} a T_{max} jsou měřením minima a maxima RTT testované při přenosu
- $B(k+1)$ je měřením maxima dosažené propustnosti během poslední události přetížení

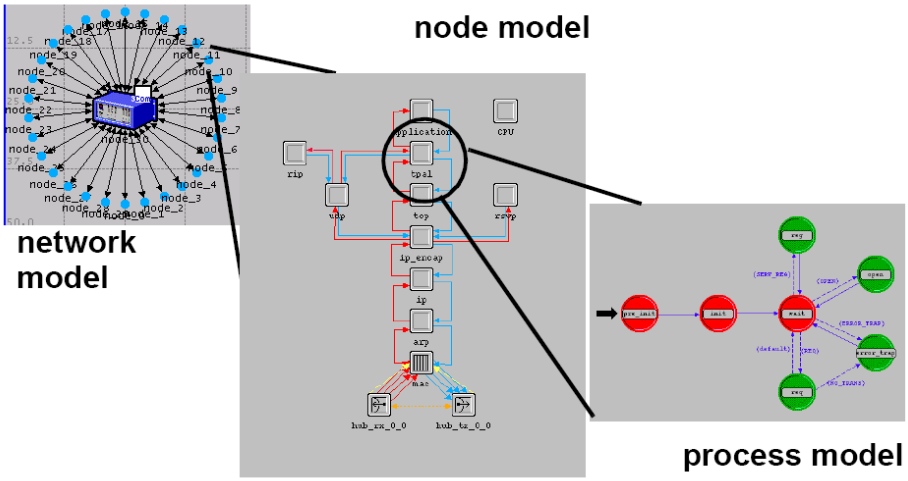
5 SÍŤOVÉ SIMULAČNÍ NÁSTROJE

Pro simulaci výkonnostních problematik TCP protokolu se dá vybírat ze dvou možných simulačních nástrojů, a tím je Opnet Modeler a Network Simulator. Opnet Modeler díky svojí rozsáhlé knihovně podporující množství protokolů, aplikací, rozličných síťových modelů a v neposlední řadě dostupností v počítačových učebnách Utko (Opnet Modeler je komerční program, který je v plném rozsahu možno spustit pouze po zakoupení licence) a Network Simulator díky tomu, že je volně šiřitelný, má dostupný zdrojový kód a díky tomu má velkou komunitu síťových expertů, kteří implementují nové moduly pro nové síťové protokoly a standardy. Ovšem návrh a simulace je v těchto simulačních prostředích naprosto rozdílný a každý používá jiný operační systém ke svému spuštění. Představíme si oba simulační nástroje, s popisem možností testovaní problematiky TCP výkonnosti pro naše účely.

5.1 OPNET MODELER

Opnet Modeler je softwarové prostředí pro návrh a analýzu velkého množství komunikačních sítí. Patří do celkového softwarového balíku OPNET (Optimum Network Performance) od americké firmy OPNET Technologies. Obsahuje široké možnosti v oblasti simulace a analýzy výsledků. OPNET Modeler již v sobě obsahuje řadu knihoven jednotlivých síťových

komponent, které mají dostupné kódy a přesně popisuje spojení struktur, sítí a protokolů. Budování modelu sítě je založeno na součinnosti několika editorů (editor projektu, editor uzlu a editor procesů na nejnižší úrovni zapsaný v jazyce C++), které modelují strukturu sítě, uzly sítě a vnitřní uspořádání až po jednotlivé komunikační entity či vrstvy síťového modelu a aplikace. K sestavení sítě slouží funkce Drag and Drop (vezmi a pusť).



Obr. 5.1 - Editory síťového simulátoru Opnet Modeler

Opnet Modeler potřebuje mít nainstalovaný ke svému spuštění některý překladač C/C++, spouští se pod operačními systémy Microsoft Windows, Solaris, Red Hat Enterprise Linux nebo Fedora Linux.

Výhodou tohoto simulačního nástroje je jeho jednoduchost při tvorbě žádané sítě funkcí drag and drop, dobře zpracovaný manuál a předváděcí tutoriály, objemné knihovny s dostupnými zdrojovými kódy, značné urychlení simulace a jednoduché generování rozsáhlých statistik, které je možno uložit do tabulek nebo ve formátech XML/HTML. Velikou nevýhodou je však jeho komerční pojetí a nedostupnost z důvodu zakoupení licence, díky čemuž se většina nových síťových standardů a protokolů raději implementuje a simuluje v síťovém prostředí Network Simulator NS2.

5.2 PROTOKOL TCP V PROSTŘEDÍ OPNET MODELER

Dostupné verze protokolu TCP v Op.Mod. verze 14.0
Tahoe
Reno
New Reno
Sack

Tab. 5.1 - Verze protokolu TCP v OM

Nastavitelné TCP parametry v Op.Mod.verze 14	
MSS	(B)
Receive Buffer	(B)
Maximum ACK delay	(s)
Maximum ACK segment	(1-x)
Slow-start Initial Count	(*MSS*) (1-4)
Fast Retransmit	(Enabled, Disabled)
Duplicate ACK Threshold	(1-x)
Fast Recovery	(Enabled, Disabled)
Selective ACK(SACK)	(Enabled, Disabled)
ECN Capability	(Enabled, Disabled)
Nagle Algorithm	(Enabled, Disabled)
Karns algorithm	(Enabled, Disabled)
Minimum, Maximum RTO	(s)
RTT Deviation Coefficient	(s)

Tab. 5.2 - Nastavitelné parametry protokolu TCP v OM

5.3 NETWORK SIMULATOR NS2

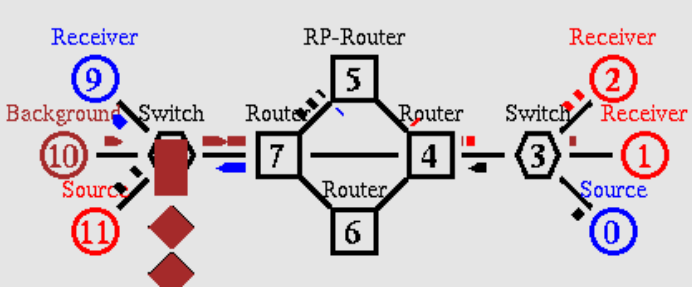
Network Simulator je volně šiřitelný síťový simulátor, který má otevřenou architekturu, která umožňuje uživatelům přidávat nové funkce. NS 2 je objektově řízený simulátor sítě, řízený diskrétními událostmi. Byl vyvinut na univerzitě v Berkeley v programovacím jazyku C++ (pro operace související s daty) a OTcl (pro operace související s řízením). Základní operace, které NS2 umožňuje jsou:

- budování modelu sítě
- stanovení cesty v modelu sítě
- budování spojení (na úrovni vyšších vrstev OSI modelu)
- generování provozu
- modelování poruch
- monitorování provozu

Budování modelu se provádí vytvořením/editací textového souboru v jazyce Tcl. Znázornění výsledků probíhá nezávisle na simulačním nástroji NS2. Pro grafické znázornění definované

topologie a jednotlivých generovaných provozů během simulace slouží nástroj Nam (Network Animator)

Nástroj NS2 byl vyvinut především pro unixovské platformy. Ke spuštění na platformě Win 32 je nutné využít emulátor unixovského prostředí – Cygwin.



Obr. 5.2 - Vytvořená architektura sítě zobrazená v prostředí Network Animator

5.4 ZPŮSOB VYTVÁŘENÍ TOPOLOGIE SÍTĚ A NASTAVENÍ SLEDOVANÝCH CHARAKTERISTIK V PROSTŘEDÍ NS2

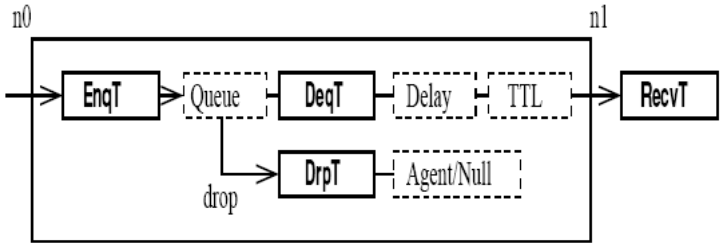
Pokud chce uživatel NS2 vytvořit jednoduchý model sítě, musí napsat tcl skript, který poté spustí v příkazové řádce unixového prostředí. Ve skriptu první co musí udělat, je deklarovat výběr směrovacího protokolu. Poté musí definovat zdrojové a cílové uzly, tyto propojit linkami, definovat agenty aplikačního a transportního protokolu, které jsou připojeny na zdrojových uzlech a konzumenty těchto provozů připojené na straně příjemců.Nakonec nastaví uživatel celkovou dobu simulace a časy spouštění jednotlivých aplikací.

Jako konzument paketů se může použít například **Loss Monitor Agent**. Je to konzument paketů, který uchovává statistiky o přijatém provozu, jako je množství přijatých a ztracených informací. Výstup těchto informací se ukládá do textového souboru. S použitím agentu Loss Monitor můžeme získat následující informace:

- *nlost_* počet ztracených paketů
- *npkts_* (počet přijatých paketů
- *bytes_* (počet přijatých bajtů
- *lastPktTime_* čas ve kterém byl přijat poslední paket
- *expected_* očekávané sekvenční číslo dalšího paketu

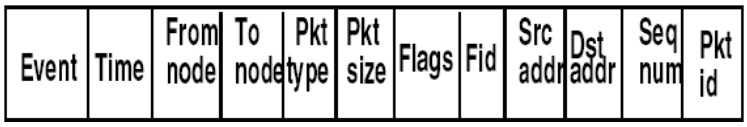
Trasování v NS 2

Pro měření zpoždění může být využita vlastnost NS2 zapisovat všechny události evidované v síti do trasovacího souboru ve formátu ascii. Pokud použijeme trasování, NS2 vloží 4 objekty do linky mezi uzly: *EnqT, DeqT, RecvT a DrpT*, jak je zobrazeno na obrázku č.XX. *EnqT* zaznamenává informace o paketech, které jsou řazeny do vstupní fronty linky. Pokud je paket vyhozen z důvodu zaplnění fronty je tato informace zaznamenána pomocí *Drp*T. *DeqT* zaznamenává, pokud je paket poslán z fronty. Nakonec *RecvT* ná podává informaci, že byl paket přijat na výstupu linky.



Obr. 5.4 - Trasovací objekty v lince

Pokud použijeme trasování do výstupního souboru ve formátu ascii, zápis je organizován do 12 polí jak je zobrazeno na obrázku č.14.



Obr. 5.5 - Jednotlivá pole v jednom řádku výstupu trasovacího souboru

Významy těchto polí jsou:

1) Typ události. Mohou zde být čtyři různé symboly: r,+,-,d které znamenají: přijaty paket (na výstupu linky), zařazení do fronty, vyslání z fronty a zahození paketu.

2) Čas, ve kterém nastala událost

3) Vstupní uzel linky

4) Výstupní uzel linky

5) Typ paketu (například CBR, UDP, TCP)

6) Velikost paketu

7) Příznaky

8) Identifikátor datového toku

9) Zdrojová adresa ve formátu uzel.port

10) Cílová adresa ve formátu uzel.port

11) Sekvenční číslo paketu síťové vrstvy

12) Poslední pole uchovává unikátní identifikátor id paketu

Příklad takového trasovacího souboru je uveden níže.

+ 8.722667 11 8 cbr 200 ------- 2 11.1 -2147483648.0 1448 2769

- 8.722667 11 8 cbr 200 ------- 2 11.1 -2147483648.0 1448 2769

r 8.722768 10 8 cbr 512 ------- 0 10.1 1.2 941 2764

+ 8.722768 8 7 cbr 512 ------- 0 10.1 1.2 941 2764

d 8.722768 8 7 cbr 512 ------- 0 10.1 1.2 941 2764

5.5 PROTOKOL TCP V PROSTŘEDÍ NS2

Nastavitelné TCP parametry v základní balíku NS2	
window_ 10000	maximální velikost okénka *cwnd*
windowInit_ 1	počáteční hodnota okénka *cwnd*
windowOption_ 1	algoritmus pro předcházení zahlcení
windowConstant_ 4	používá se jen pokud windowOption != 1
windowThresh_ 0.002	požívá se pro výpočet průměru okna
overhead_ 0	přidává náhodný čas mezi posíláním
ecn_ 0	reakce na nastavení *ECN* bitu
packetSize_ 1460	velikost paketu stanovená vysílačem (B)
maxrto_ 64	hranice časovače *RTO* (s)
dupacks_ 0	čítač duplikátních ACK
ack_ 0	nejvyšší počet přijatých ACK
cwnd_ 0	okénko zahlcení *cwnd*
ssthresh_ 0	slow-start práh
rtt_ 0	*RTT* vzorek
srtt_ 0	vyhlazená (průměrovaná) hodnota *RTT*
rttvar_ 0	střední odchylka *RTT*
backoff_ 0	současný činitel *RTO* backoff
maxseq_ 0	max. poslané sekvenční číslo

Tab. 5.3 - Nastavitelné parametry protokolu TCP v NS2

Dostupné verze protokolu TCP v NS2		
Základní balík NS2	Dostupné moduly	
Tahoe	HighSpeed	Fast - TCP
Reno	H-TCP	TCP Low-Priority
New Reno	Scalable TCP	TCP Hybla
Sack	Westwood	Rate-Halving TCP
Vegas	Veno	
Fack	Compound	

Tab. 5.4 - Verze protokolu TCP v NS2

5.6 VLASTNOSTI TCP V NS2

- NS2 eviduje všechny pakety přenášené v síti v operační paměti. Při simulaci vysokorychlostních sítí je třeba omezit počet hlaviček, které budou pro každý paket evidovány
- TCP neimplementuje řízení toku proti zahlcení přijímače
- TCP implementuje řízení zahlcení proti zahlcení sítě. Okénko zahlcení může růst až do limitu specifikovaného proměnnou *window_* TCP vysílače

6 TESTOVÁNÍ TCP VARIANT PRO VYSOKORYCHLOSTNÍ SÍTĚ V NS2

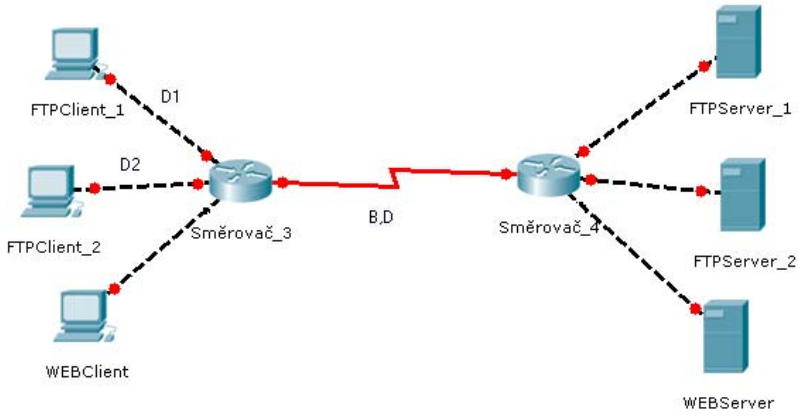
6.1 TOPOLOGIE A PARAMETRY SÍTĚ

Pro účely testování přenosových charakteristik TCP variant definujeme topologii sítě na obrázku 6.1. Síť zahrnuje dva FTP klienty, a dva FTP servery. V některých simulacích bude také webový klient navazovat definované relace s web serverem, což nám zajistí, že sdílenou linku nebudou využívat pouze FTP přenosy, ale i jiná aplikace, což lépe odpovídá reálné situaci v síti. Klienti FTP budou stahovat po definovanou dobu data ze serverů, budou se snažit zabrat si co největší šířku pásma sdílené linky.

Tato síť je dostačujícím příkladem situace, kdy FTP přenosy, využívající ke svému spolehlivému přenosu protokol TCP, sdílí po cestě stejnou přenosovou linku, která je úzkým

hrdlem sítě. Šířku pásma sdílené linky B budeme nastavovat v rozmezí od 10Mb/s - 1Gb/s. Sdílené lince definujeme zpoždění D. Přístupové klientské linky budou mít vždy dvojnásobnou kapacitu, než linka sdílená. Zpoždění těchto linek, D1 a D2, budeme měnit v rozmezí od 16ms do 160ms. Tím tak zajistíme podmínky, kdy přenosy vstupující na sdílenou linku mají rozdílné časy RTT. Tímto můžeme sledovat férovost jednotlivých přenosů, to znamená, jak si rozdělí dostupnou šířku pásma sdílené linky.

Velikost vyrovnávací paměti Q pro sdílenou linku na směrovači tři se mění podle veličiny BDP. To znamená, že pokud například nastavíme RTT D1 = 160ms a sdílenou linku B = 1Gb/s, maximální kapacita paměti bude potom Q = B*D = 20MB = 13333 paketů (kdy 1 paket = 1500B). Tato fronta při některých simulacích nebude stále stejná, mění se v poměru ke Q. Fronty směrovačů jsou typu DropTail. Vyrovnávací paměť přijímače nebude neomezená, doporučuje se nastavit alespoň velikost podle BDP. V našem případě definujeme pro všechny simulace 15000 paketů, to je 22,5 MB. Dobu měření zvolíme 10 minut, přičemž vždy první FTP aplikace využívá standardní TCP algoritmus a zahajuje stahování po 100s a druhá následuje v době 200s využívající některý vysokorychlostní TCP.



Obr. 6.1- Definovaná topologie sítě

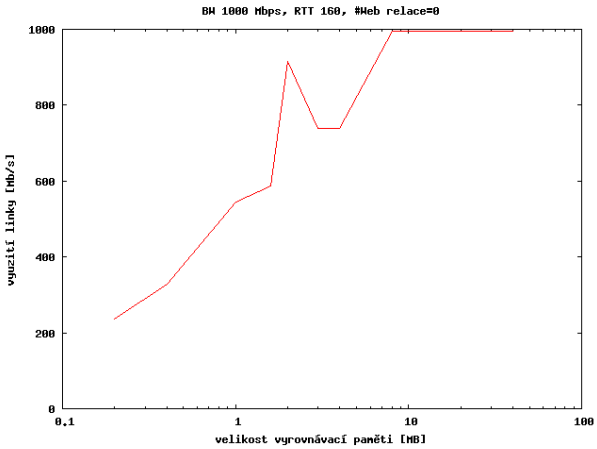
6.2 VÝKONNOST

Jako první budeme testovat výkonnost jednotlivých TCP variant. Výkonnost znamená využití dostupných síťových zdrojů. Je známo, že výkonnost standardního TCP je ovlivněna velikostí vyrovnávací paměti prvků v síti. Pro jeden TCP přenos (nebo více synchronních přenosů) se snižuje využití sítě, pokud je velikost fronty směrovače menší, než daný BDP pro konkrétní linku. Výkonnost také ovlivňuje množství soutěžících přenosů. Protože podmínky v síti

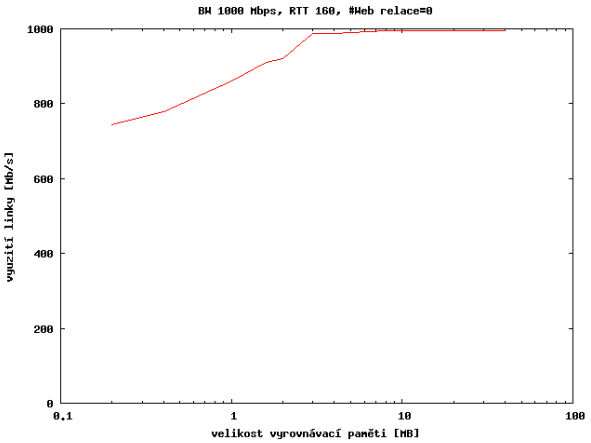
nejsou stále statické, bude nás zajímat schopnost rychle získat nebo uvolnit šířku pásma při změnách podmínek, například pokud se spustí nebo zastaví přenos.
Abychom mohli vyhodnotit výkonnost (využití linky), budeme uvažovat dva TCP toky, které mají stejné zpoždění šíření signálu.

6.2.1 VÝKONNOST V ZÁVISLOTI NA VELIKOSTI VYROVNÁVACÍ PAMĚTI SMĚROVAČE

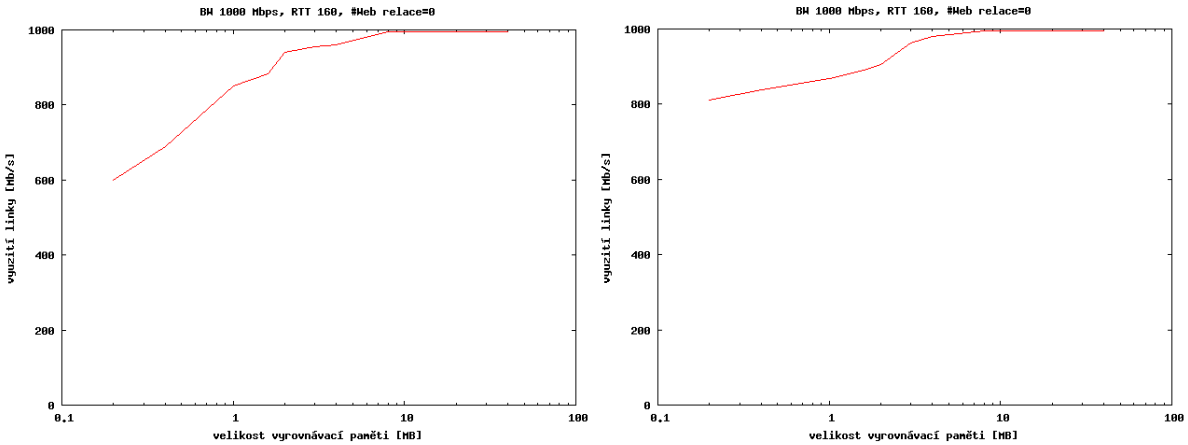
Budeme měřit využití linky v závislosti na velikosti fronty směrovače. Velikost fronty se bude měnit od 1% BDP do 200% BDP. Na osu x vynášíme v MB, velikost fronty bude v rozmezí 200kB - 40MB. Sdílené lince nastavíme šířku pásma BW=1Gb/s, na linku budou vstupovat postupně dva FTP přenosy a stejných RTT = 160ms. V síti neprobíhají žádné webové relace. Jak je vidět z obrázků 6.2, nové protokoly dosahují lepší výkonnosti, než standardní TCP. Je to dáno jejich vlastnostmi, díky kterým si poradí se zahazováním paketů na směrovači.
V tomto případě Reno dosahuje plné využití linky až od 8MB, zatímco Highspeed již od 2MB vykazuje přes 90% využití linky. V tomto testu dopadl nejlépe HTCP, který i při nejmenších velikostech paměti dosahuje téměř 80% využití linky.



Reno



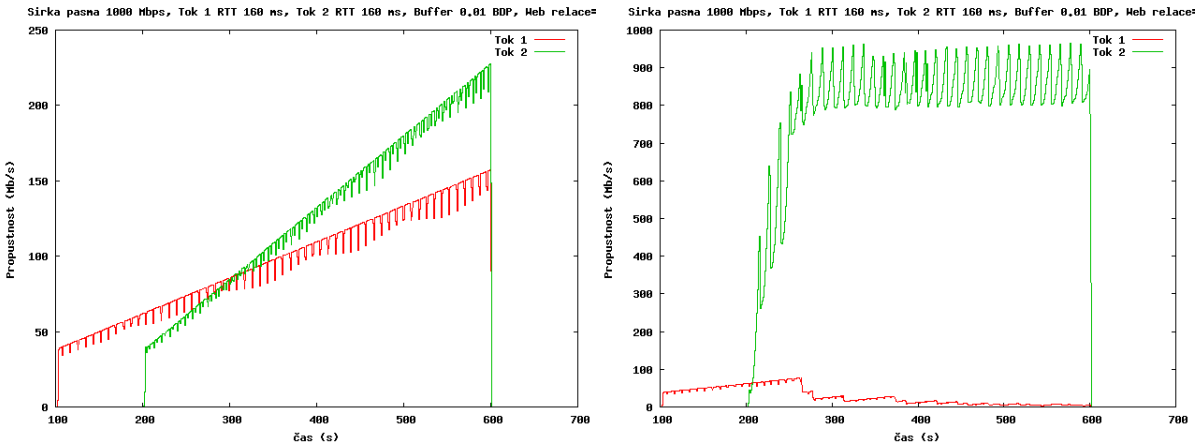
Scalable



Highspeed HTCP

Obr.6.2 - Využití linky v závislosti na vyrovnávací paměti směrovače

Na obrázku 6.3 je zobrazen konkrétní případ propustnosti obou přenosů v závislosti na čase pro varianty Reno a HTCP. Velikost fronty směrovače je v tomto případě 1% BDP, to je 200kB. Z grafů je vidět, že Reno velice pomalu zabírá šířku pásma sdílené linky, po 8 minutách je přibližně na 1/3 dostupné kapacity. Přenos využívající HTCP, který vstupuje později na sdílenou linku, sice zabere dostupnou kapacitu, ovšem téměř znemožní přenos dřívějšího FTP sezení.



Reno HTCP

Obr. 6.3 - Propustnosti přenosů v závislosti na čase, BW = 1Gb/s, RTT1 = 160ms, RTT2 = 160ms, Fronta = 1% BDP, Web relace=0

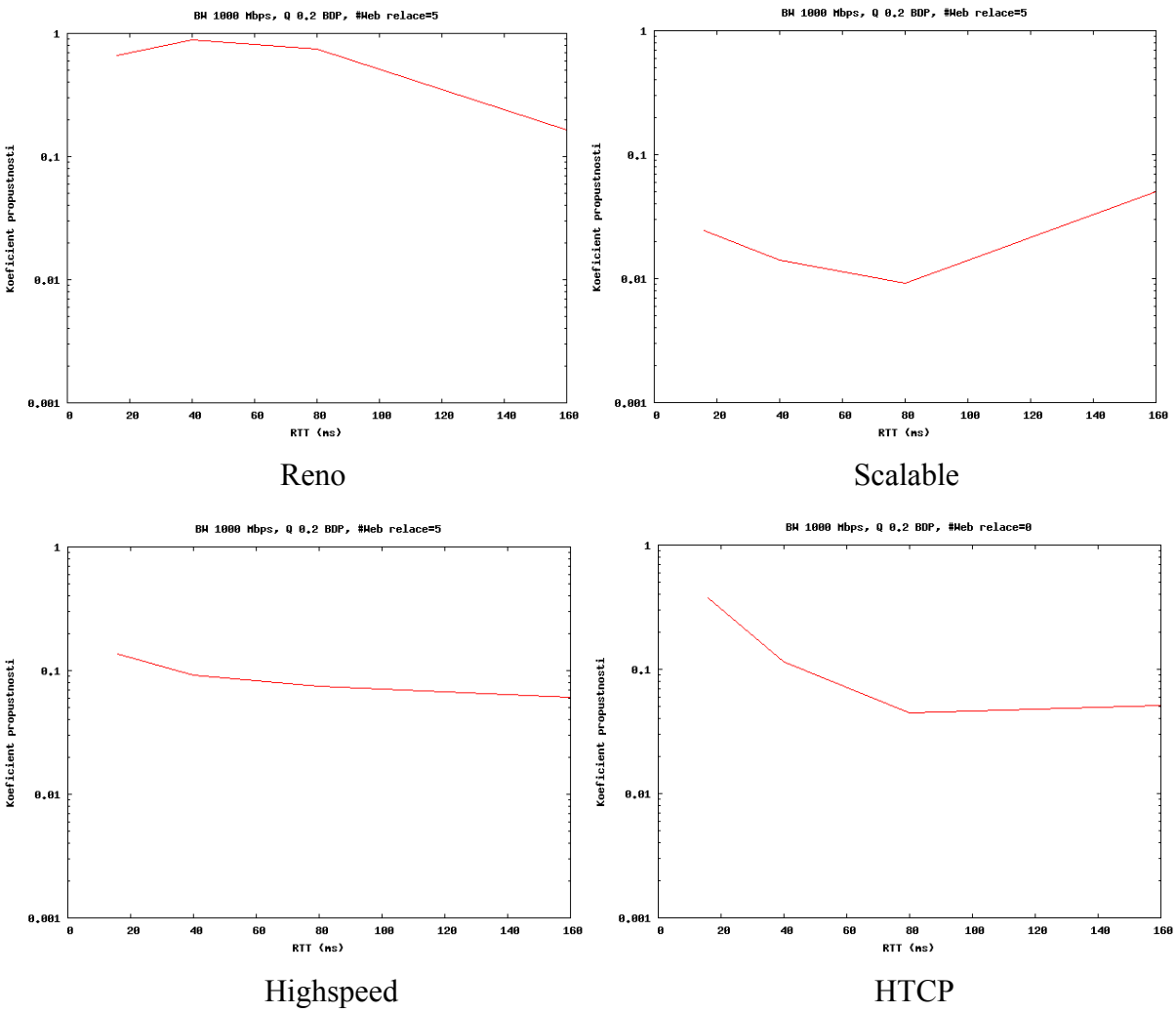
6.3 FÉROVOST

Tento test bude zkoumat dopad RTT jednotlivých soutěžících toků na jejich propustnost. V síti s jedním úzkým hrdlem budeme předpokládat, že soutěžící dlouhotrvající FTP přenosy se stejnými časy RTT by měly dosahovat přibližně stejné propustnosti. Přenosy s odlišnými časy RTT by měly být k sobě nefér. Pokud bude použit standardní algoritmus vyhýbání se zahlcení TCP, toky s kratšími RTT budou obecně dosahovat větší propustnosti než toky s delšími RTT.

6.3.1 FÉROVOST PŘI SYMETRICKÝCH PŘENOSOVÝCH PODMÍNKÁCH

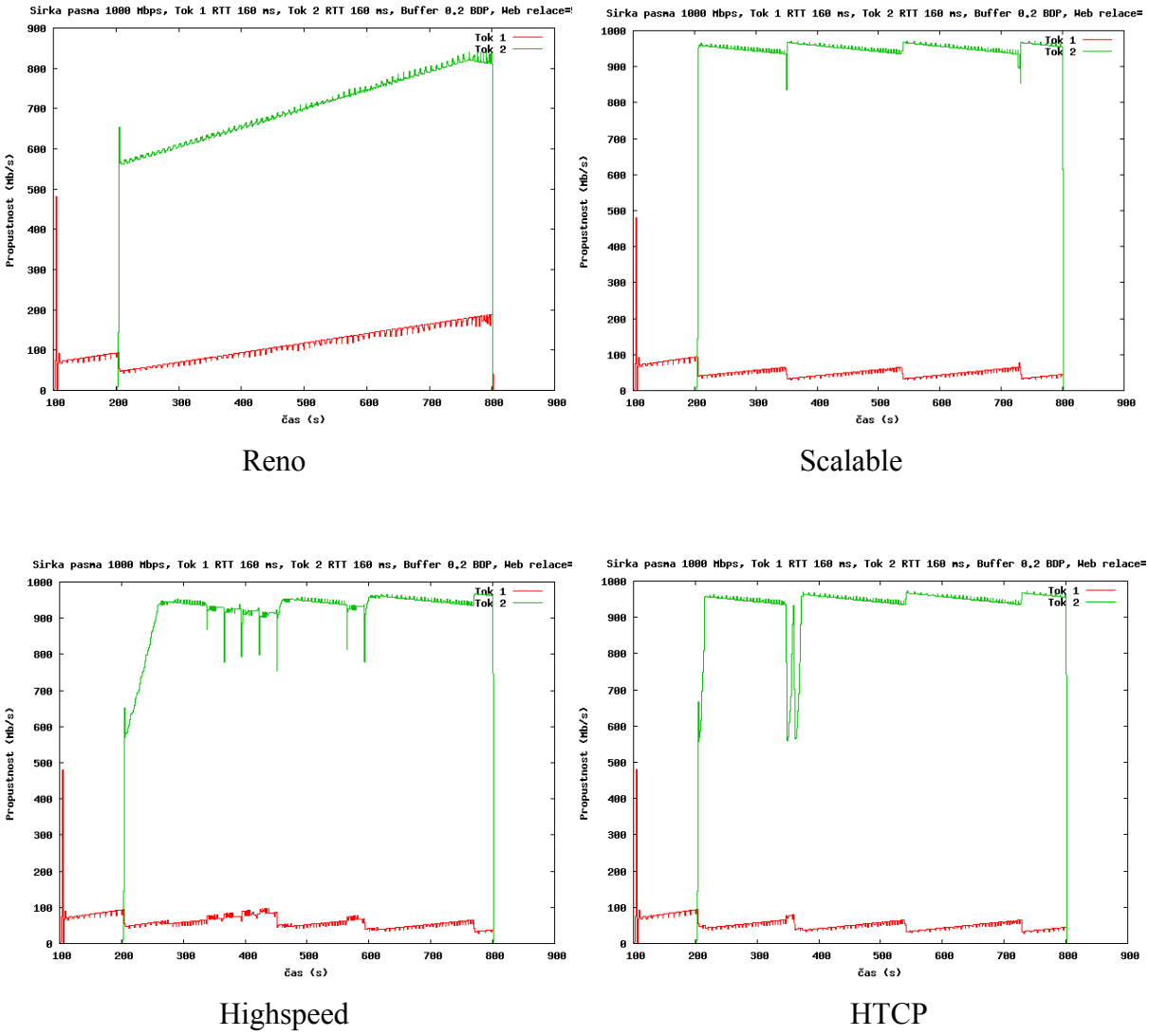
Budeme měřit průměrnou propustnost obou přenosů a koeficient propustnosti při symetrických přenosových podmínkách v síti, to znamená, že každý individuální přenos má stejné zpoždění šíření signálu a sdílí jednu linku, která je úzkým hrdlem. Měření budeme vyhodnocovat pro řadu rozsahů šířky pásma sdílené linky. (10Mb/s, 100Mb/s, 1000Mb/s). Zpoždění šíření signálů linek D1 a D2 bude vždy stejně definované, měnící se současně od 16ms do 160ms. Velikost vyrovnávací paměti směrovače bude konstantní poměr veličiny BDP, tedy výsledek vynásobení šířky pásma a zpoždění linky, zvolíme 20% a 100% BDP, který přibližně odpovídá podmínkám malých a velkých front v síti.

Na obrázku 6.4 je zobrazena závislost koeficientu propustnosti pro oba přenosy na řadě RTT obou FTP relací. Koeficient propustnosti je poměrem přenosových rychlostí obou FTP relací. V tomto případě má linka šířku pásma 1Gb/s a velikost fronty vyrovnávací paměti směrovače je 20% BDP. Měření je pro dva FTP přenosy a pět webových relací. Na osu y budeme vynášet v logaritmickém měřítku koeficient propustnosti sdílené linky v závislosti na RTT obou přenosů. Jak je vidět z grafů, měření se liší pro zvolené varianty TCP. Reno si pro menší časy RTT zachovává poměr rychlostí blížící se k jedné, to znamená, že oba přenosy mají přibližně stejné rychlosti. Pro 160ms je ale koeficient rovný přibližně 0,1. Vysokorychlostní varianty se drží přibližně na hodnotě 0,1. Nejhůře v tomto případě dopadl Scalable TCP a nejlépe HTCP, který se při 16ms obou FTP relací přiblížil k hodnotě 0,4.



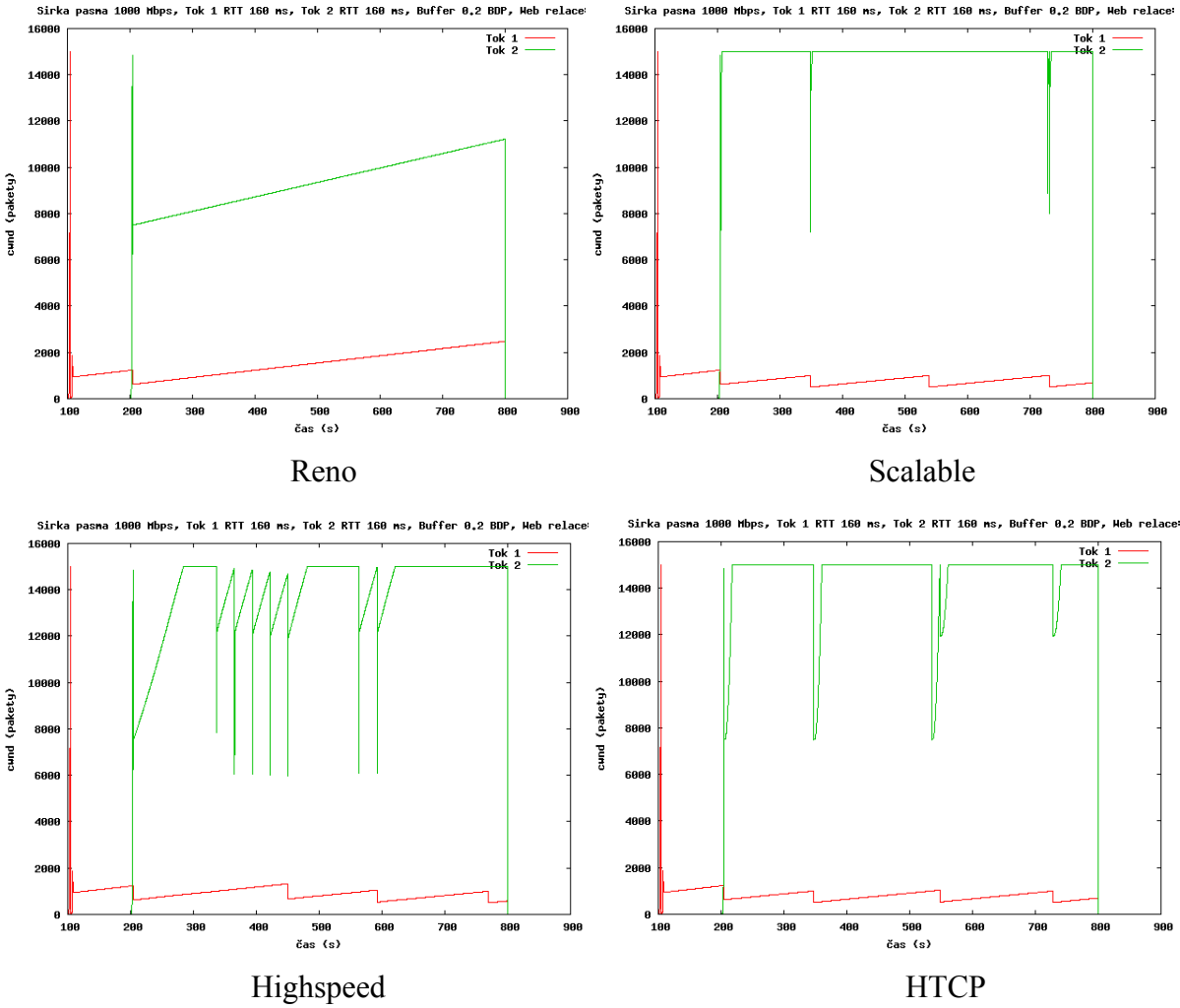
Obr. 6.4 - Závislost koeficientu propustnosti na RTT sdílených přenosů, BW = 1Gb/s, Fronta = 20% BDP, Web relace=5

Na obrázku 6.5 je zobrazena konkrétní závislost propustností obou přenosů na čase. Šířka pásma sdílené linky je 1Gb/s, RTT obou přístupových linek je 160ms a vyrovnávací paměť směrovače je nastavena na 20% BDP. Na linku zároveň vstupuje provoz z webových relací. Podle předchozího obrázku XX by se koeficient propustnosti měl pohybovat pro všechny případy kolem 0,1. Z grafů propustností je vidět, že pro všechny varianty Reno na počátku trpí nedostatkem vyrovnávací paměti na směrovači, i na volné lince jeho rychlost padá přibližně ke 100Mb/s. Nejvíce férově se v tomto případě chová samotný Reno, nejméně Scalable TCP.



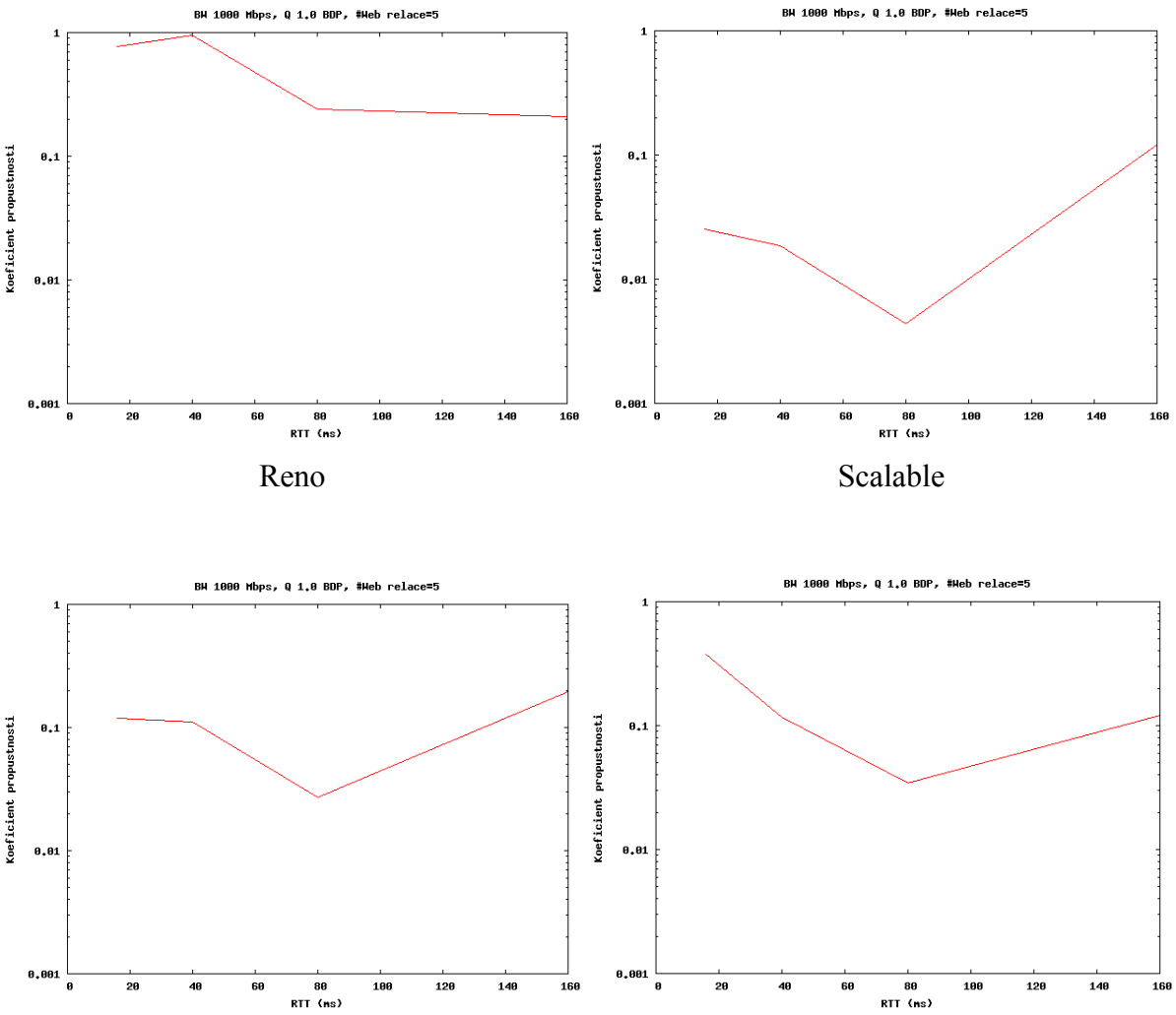
Obr. 6.5 - Závislosti propustností na čase, BW = 1Gb/s, RTT1 = 160ms, RTT2 = 160ms, Fronta = 20% BDP, Web relace=5

Na obrázku 6.6 je zobrazena konkrétní závislost okénka zahlcení cwnd obou přenosů na čase. Šířka pásma sdílené linky je 1Gb/s, RTT obou přístupových linek je 160ms a vyrovnávací paměť směrovače je nastavena na 20% BDP. Na linku zároveň vstupuje provoz z webových relací. Z grafů lze pozorovat, že okénko zahlcení cwnd je limitováno okénkem přijímače rwnd, roste až do definované hodnoty 15000 paketů, což odpovídá 22,5MB. Pro Reno poté prudce klesá a pomalu se lineárně zvětšuje, čímž postupně narůstá i přenosová rychlost. Lze pozorovat, že pro vysokorychlostní varianty se okénko navyšuje během několika sekund, svým agresivním navyšováním ovšem znevýhodní dřívější přenosy.



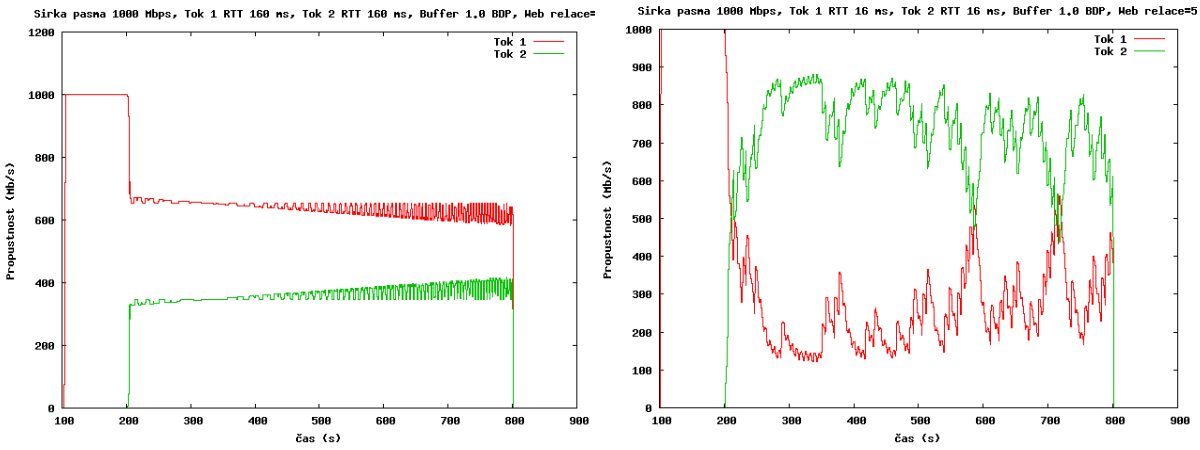
Obr. 6.6 - Závislosti okénka zahlcení cwnd na čase, BW = 1Gb/s, RTT1 = 160ms, RTT2 = 160ms, Fronta = 20% BDP, Web relace=5

Na obrázku 6.7 je zobrazena závislost koeficientu propustnosti pro oba přenosy na řadě RTT obou FTP relací. V tomto případě má linka šířku pásma 1Gb/s a velikost fronty vyrovnávací paměti směrovače je nyní 100% BDP. Z grafů je vidět, že nejlépe na tom je Reno, jehož přenosy si pro kratší časy RTT téměř rozdělí dostupnou šířku pásma. Vysokorychlostní varianty se i pro dostatečné fronty směrovače chovají neférově, jako v předchozím případě nejhůře dopadl Scalable TCP a nejlépe HTCP. Při dalších simulacích, s pamětmi směrovače rovné dvojnásobku BDP, férovost se pro vysokorychlostní varianty příliš nezvyšuje. Lepších výsledků pro vyšší fronty dosahují přenosy s delšími časy RTT.



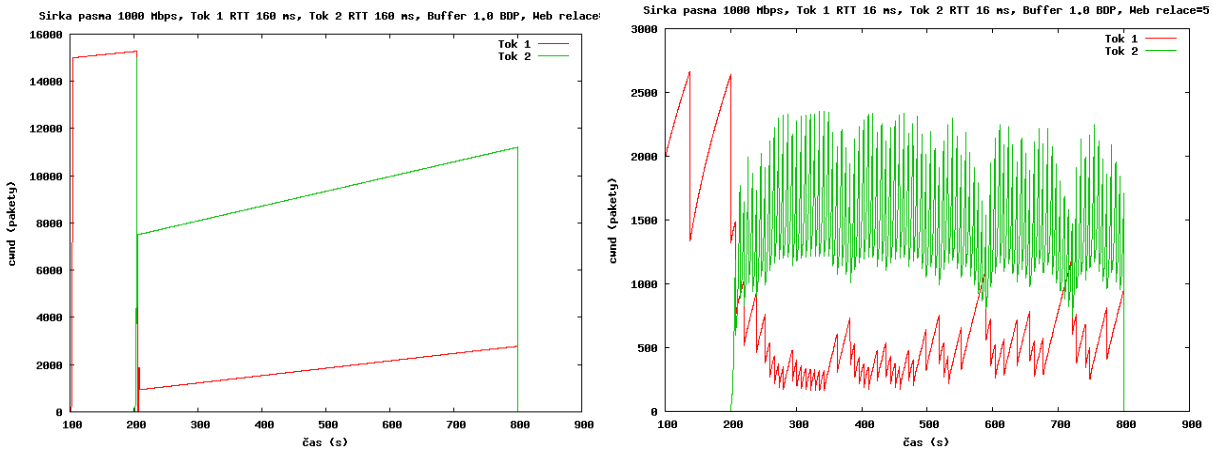
Obr. 6.7 - Závislost koeficientu propustnosti na RTT sdílených přenosů, BW = 1Gb/s, Fronta = 100% BDP, Web relace=5

Na obrázku 6.8 je zobrazena konkrétní závislost propustností obou přenosů na čase a na obrázku 6.9 závislosti okénka zahlcení cwnd na čase pro Reno (RTT = 160ms) a HTCP (RTT = 16ms). Šířka pásma sdílené linky je 1Gb/s, vyrovnávací paměť směrovače je nastavena na 100% BDP. Na linku zároveň vstupuje provoz z webových relací. Z grafů lze pozorovat, že Reno při dostatku paměti zabírá na počátku celou dostupnou šířku pásma sdílené linky, po vstupu dalšího přenosu si oba přenosy téměř rozdělí dostupnou šířku pásma. HTCP se při 16ms pro oba přenosy chová férověji než ostatní vysokorychlostní varianty. U okénka zahlcení cwnd pro Scalable TCP lze pozorovat agresivnější nárůst při ztrátovosti, přenos vstupující první na linku využívající Reno již není úplně zahlušen.



Reno HTCP

Obr. 6.8 - Závislosti propustností na čase, BW = 1Gb/s, Reno (RTT1 = 160ms, RTT2 =160ms), HTCP (RTT1 = 16ms, RTT2 =16ms) Fronta = 100% BDP, Web relace=5



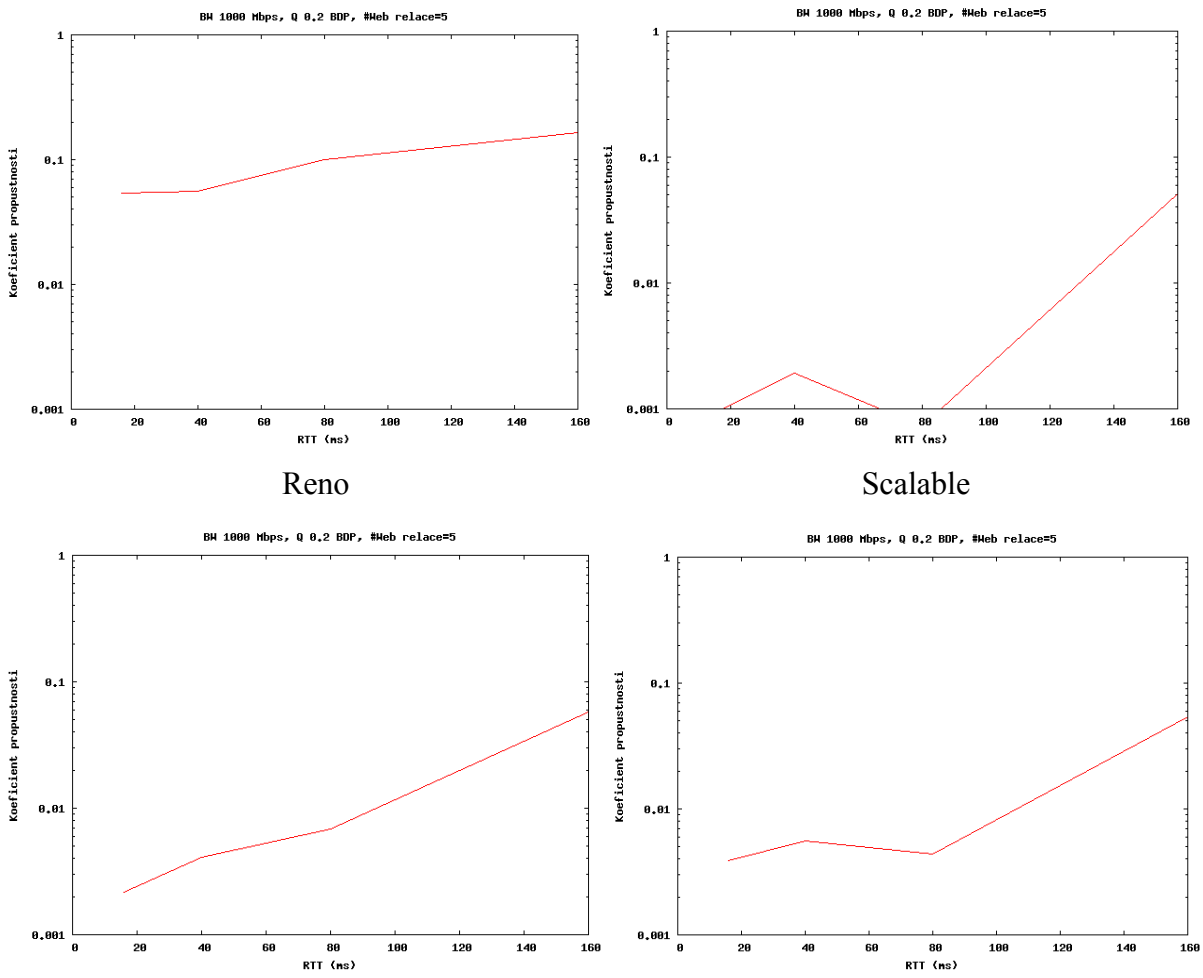
Reno HTCP

Obr. 6.9 - Závislosti okénka zahlcení cwnd na čase, BW = 1Gb/s, Reno (RTT1 = 160ms, RTT2 =160ms), HTCP (RTT1 = 16ms, RTT2 =16ms), Fronta = 100% BDP, Web relace=5

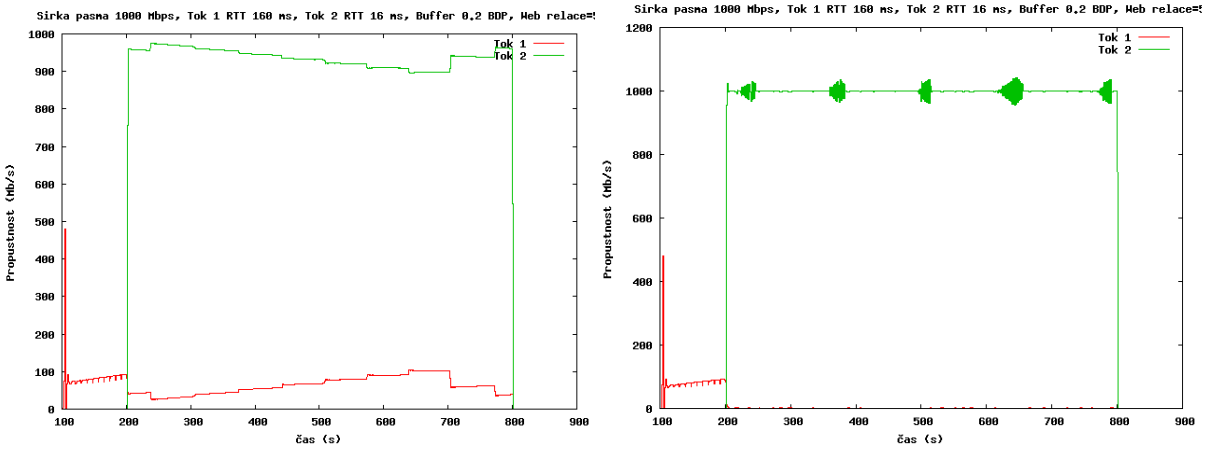
6.4 FÉROVOST PŘI ASYMETRICKÝCH PŘENOSOVÝCH PODMÍNKÁCH

Při tomto testu se zaměříme na průměrnou propustnost v závislosti na čase a koeficient propustnosti, pokud je zpoždění šíření signálu prvního přenosu konstantní (nastavíme 160ms) a zpoždění šíření signálu druhého přenosu se mění v rozmezí 16ms až 160ms. Měření budeme vyhodnocovat pro řadu rozsahů šířky pásma sdílené linky (10Mb/s, 100Mb/s, 1000Mb/s) a řadu zpoždění šíření signálu druhého přenosu. Velikost vyrovnávací paměti linky bude konstantní poměr veličiny BDP. Výsledky měření jsou zobrazeny jako v předchozím případě pro linku s šířkou pásma 1Gb/s a velikost fronty vyrovnávací paměti směrovače 20% BDP a 100% BDP. Výsledky testování pro nižší šířky pásma sdílené linky jsou přiloženy na CD.

Na obrázku 6.10 je zobrazena závislost koeficientu propustnosti pro oba přenosy na řadě RTT FTP relace, která vstupuje později na linku. V tomto případě má sdílená linka šířku pásma 1Gb/s a velikost fronty vyrovnávací paměti směrovače je 20% BDP. Měření je pro pět webových relací. Z obrázků lze rozeznat, že ani Reno není pro menší fronty směrovače příliš férový. Čím menší má pozdější FTP relace čas RTT, tím méně je sdílení linky férovější. Pro Reno se pohybuje na hranici 0,1. Vysokorychlostní varianty pro nižší časy RTT druhého přenosu první přenos téměř zahluší, přičemž nejhůře v tomto testu skončil Scalable TCP. Se vzrůstajícím RTT roste i férovost, která nedosáhne více než 0,1. Na obrázku 6.11 a 6.12 jsou zobrazeny příklady přenosových rychlosti a okénka zahlcení na čase, pro Reno a Scalable TCP. Přenosy vstupující na linku s časem RTT=16ms zabírají téměř celé pásmo sdílené linky, v případě Scalable TCP je ale první přenos s delším RTT úplně zahlušen. U okénka zahlcení Scalable TCP je dobře rozpoznatelná vysoká frekvence sondování dostupné kapacity linky.

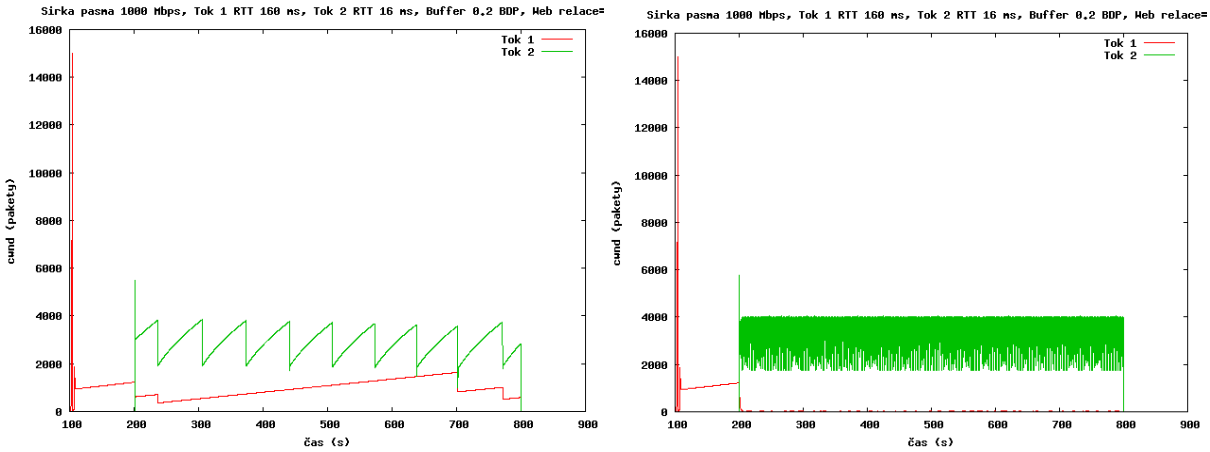


Obr. 6.10 - Závislost koeficientu propustnosti na RTT2 pozdějšího přenosu, BW = 1Gb/s, RTT1 = 160ms, RTT2 = 16 - 160ms, Fronta = 20% BDP, Web relace=5



Reno Scalable

Obr. 6.11 - Závislost propustností obou přenosů na čase, BW = 1Gb/s, RTT1 = 160ms, RTT2 = 16ms, Fronta = 20% BDP, Web relace=5

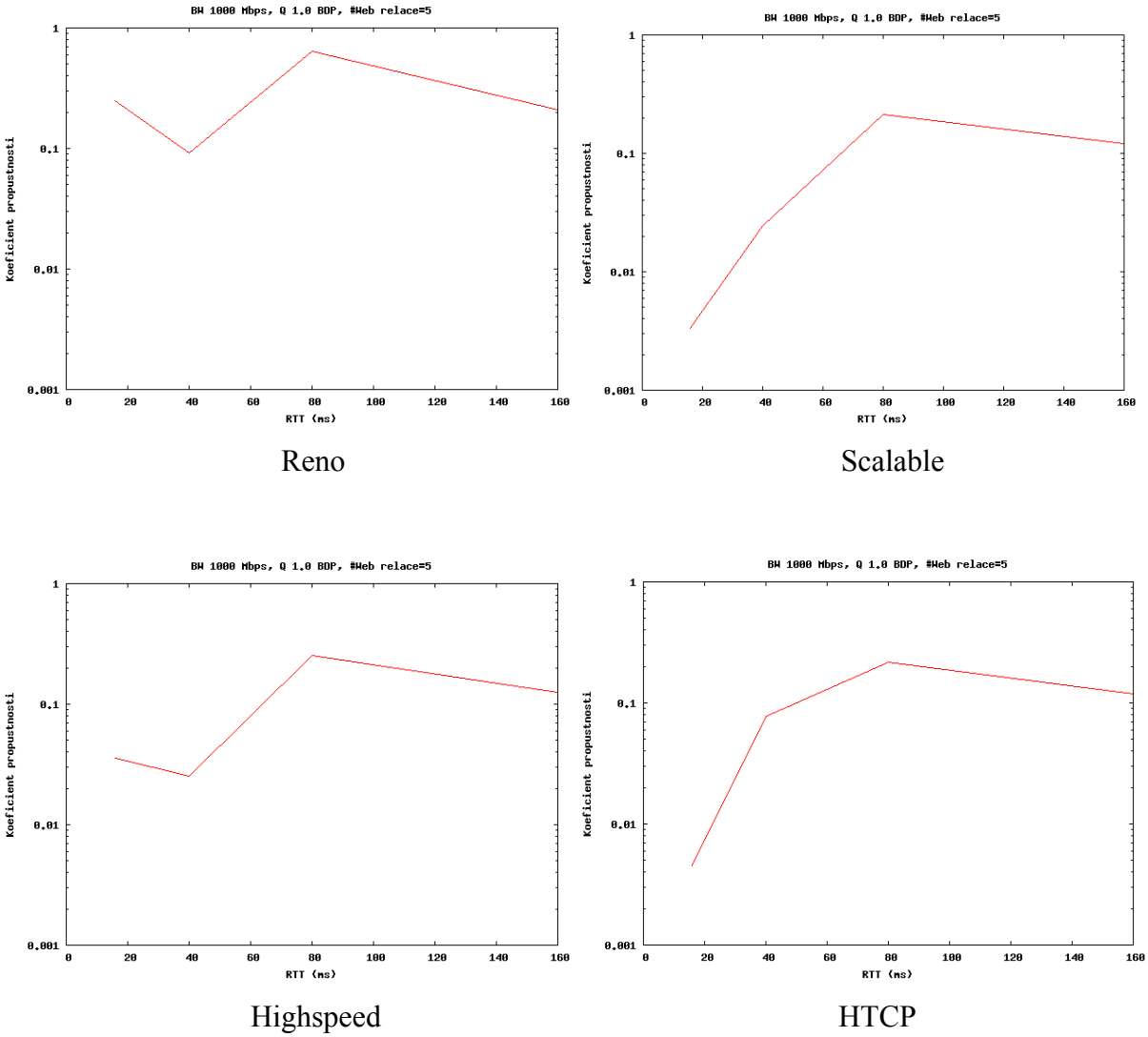


Reno Scalable

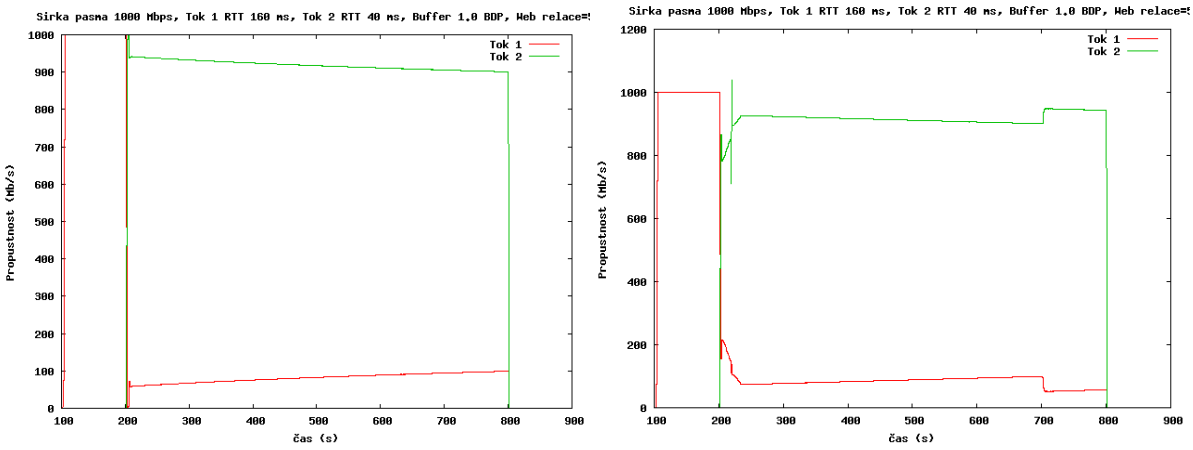
Obr. 6.12 - Závislost okének zahlcení cwnd obou přenosů na čase, BW = 1Gb/s, RTT1 = 160ms, RTT2 = 16ms, Fronta = 20% BDP, Web relace=5

Na obrázku 6.13 je zobrazena závislost koeficientu propustnosti pro oba přenosy na řadě RTT FTP relace, která vstupuje později na linku. V tomto případě má sdílená linka šířku pásma 1Gb/s a velikost fronty vyrovnávací paměti směrovače je nyní 100% BDP. Pro dostatečné fronty směrovače se férovost pro Reno zvyšuje, tedy až na případ, kdy má druhý přenos čas RTT = 40ms. S nulovým počtem webových relací na sdílené lince se férovost pro Reno zvýší téměř k hodnotě 1. V tomto testu si nevede dobře žádná z vysokorychlostních variant TCP.

Toky přicházející na linku s kratšími časy RTT jsou téměř zahlušeny. O něco lepších výsledků dosahují pro nulový počet webových relací na sdílené lince.



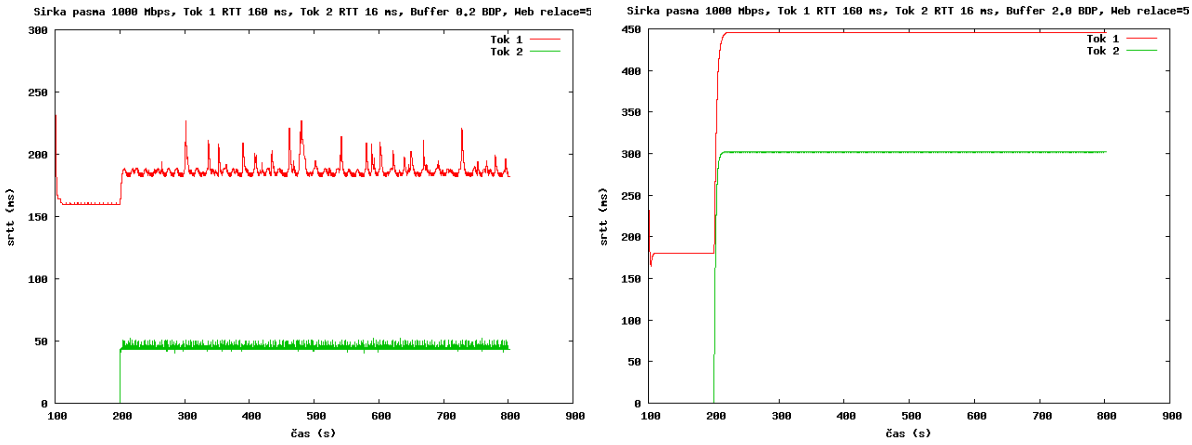
Obr. 6.13 - Závislost koeficientu propustnosti na RTT2 pozdějšího přenosu, BW = 1Gb/s, RTT1 160ms, RTT2 16 - 160ms, Fronta = 100% BDP, Web relace=5



Reno HTCP

Obr. 6.14 - Závislost propustnosti obou přenosů na čase, BW = 1Gb/s, RTT1 = 160ms, RTT2 = 40ms, Fronta = 100% BDP, Web relace=5

Jak lze pozorovat z grafů výše, ani pro fronty nastavené na 100% BDP se pro vysokorychlostní varianty příliš nezvýšila férovost sdílejících přenosových toků. Situace se výrazně zlepší, až když nastavíme frontu směrovače na dvojnásobek BDP. Se zvyšujícím se RTT druhého přenosu vstupujícího na linku, si přenosy sice férově rozdělí sdílenou linku, stoupne ale jejich celkové zpoždění z důvodu dlouhého čekání paketů ve frontách směrovače, které se může pohybovat až k 0,5 s, jak je vidět na obrázku 6.15.



Scalable - a Scalable - b

Obr. 6.15 - Závislost zpoždění přenosů na čase. BW = 1Gb/s, RTT1 = 160ms, RTT2 = 16ms, Fronta = 100% BDP (a), Fronta = 100% BDP (b), Web relace=5

7 STAV IMPLEMENTACE V SOUČASNÝCH DATOVÝCH SÍTÍCH

Správná velikost TCP okénka rwin je kritická pro dosáhnutí účinného přenosu. Důležitá část je najít tu správnou velikost. Příliš malá i příliš velká okénka mohou snížit propustnost. Dobrým odhadem je použít veličinu BDP. Kromě toho můžeme zakládat svoje odhady na pravidelném měření RTT nebo paketových ztrátách a vytvářet je dynamicky. To je také důvod, proč některé výzkumné týmy zkoumaly nové algoritmy, aby za jistých okolností vyladili chování TCP. Počínaje jádrem 2.6.13, linuxové jádro podporuje zásuvné moduly pro TCP zásobník a umožňuje přepínání mezi algoritmy v závislosti na tom, k jakému systému je připojeno. První protokol používaný na internetu byl TCP Tahoe navržený v r. 1988. Pozdější varianta byla TCP Reno, nyní široce používaný TCP protokol v síti a jeho nástupce TCP NewReno. V současné době Linuxové jádro zahrnuje následující algoritmy (jádro 2.6.19.1):

- **High Speed TCP** - algoritmus popsaný v RFC 3649. Hlavní použití je pro linky s velkou šířkou pásma a velkými časy RTT. (přibližně Gb/s a 100ms RTT)
- **HTCP** - HTCP byl navržený v Hamilton Ústavu pro přenosy, které se při událostech přetížení dokážou obnovit mnohem rychleji. Je také určený pro BDP linky a navržený zohlednit férovost soutěžících přenosů.
- **Scalable TCP** - Scalable je další algoritmus pro WAN linky s velkými BDP. Jeho cílem je rychlé zotavení po události přetížení. Toho dosahuje snížením okénka na vyšší hodnotu než standardní TCP
- **TCP BIC** - BIC je zkratkou pro binární navyšování při ochraně proti zahlcení. BIC využívá zvláštní funkci pro zvětšování okénka zahlcení. V případě paketové ztráty se okno redukuje násobným faktorem. Velikost okna těsně před a po redukci je využitá pro binární hledání nové hodnoty velikosti okénka. BIC byl využívaný jako standardní algoritmus v linuxovém jádře.
- **TCP CUBIC** - CUBIC je méně agresivní varianta BIC (ve významu, že nezabírá tolik přenosové šířky pásma soutěžících TCP přenosů, jako BIC)

Přepínání mezi rozdílnými algoritmy se provádí jednoduše zapsáním textu do /proc/ vstupu:

```
debian:~# echo "htcp" > /proc/sys/net/ipv4/tcp_congestion_control
debian:~# cat /proc/sys/net/ipv4/tcp_congestion_control
htcp
```

Seznam dostupných modulů lze zobrazit příkazem:

```
debian:~# ls /lib/modules/`uname -r`/kernel/net/ipv4/
ah4.ko        multipath_drr.ko      tcp_highspeed.ko  tcp_westwood.ko
esp4.ko       multipath_random.ko   tcp_htcp.ko       tunnel4.ko
inet_diag.ko  multipath_rr.ko       tcp_hybla.ko      xfrm4_mode_transport.ko
ipcomp.ko     multipath_wrandom.ko  tcp_lp.ko         xfrm4_mode_tunnel.ko
ip_gre.ko     netfilter             tcp_scalable.ko   xfrm4_tunnel.ko
ipip.ko       tcp_cubic.ko          tcp_vegas.ko
ipvs          tcp_diag.ko           tcp_veno.ko
```

Pokud budeme kompilovat svoje vlastní jádra, najdeme moduly v Networking - Networking options - TCP: část advanced congestion. Protože nastavení ovlivňují pouze stranu odesílatele, nemusíme si všimnout rozdílů při jejich použití. Abychom viděly pozměněné chování, musíme změřit parametry TCP přenosů.

8 ZÁVĚR

Protokol TCP je v dnešních datových sítích nejběžněji používaným transportním protokolem pro aplikační vrstvu. Původní standard vznikal v době, kdy byly řádově menší přenosové rychlosti i vzdálenosti mezi komunikujícími koncovými stanicemi. V této práci je pospán vývoj protokolu TCP od prvních standardů až po dnešní varianty, které jsou navrhnuté přizpůsobit se do sítí, na které původní specifikace nebyla připravena. Je detailně vysvětlen základní princip funkce TCP a možné problémy se kterými se setkává ve vysokorychlostních a bezdrátových sítích. Jsou popsány hlavní důvody nevyužití plné kapacity výstupních linek ve vysokorychlostních a bezdrátových sítích. Ve vysokorychlostních sítích TCP sonduje šířku pásma příliš pomalu. Lineární zrychlování růstu velikosti okénka je příliš pomalé. Také při ztrátovosti na lince, původní specifikace tento jev vyhodnocuje jako zahlcení sítě, a nutí vysílající stanici až zbytečně přespříliš zpomalit množství vysílaných dat. V bezdrátových sítích neplatí předpoklad, na kterém je postavena kontrola zahlcení TCP. V těchto sítích dochází ke ztrátám ne kvůli zahlcení sítě, ale kvůli chybovosti na lince. Vznikají tedy nové varianty, snažící se adaptovat na nové prostředí, ve kterém jsou využívány k přenosu dat. V práci jsou popsány tři nejvíce diskutované varianty TCP pro vysokorychlostní sítě, které se jeví jako nejvíce perspektivní. Široké nasazení do praxe si ovšem žádá množství praktických ověření reálného chování těchto variant TCP. Pro simulace se hodí ověřené simulační nástroje, kterými jsou Opnet Modeler a Network Simulator NS2. Opnet Modeler je kvalitním nástrojem pro takové simulace, ovšem díky jeho komerčnímu pojetí, nové varianty a standardy jsou implementovány až v době, kdy jsou používané v běžných sítích. Network Simulator NS2 je volně šiřitelný síťový simulátor, který má otevřenou architekturu, která umožňuje uživatelům přidávat nové funkce. Nové standardy jsou do tohoto prostředí implementovány ještě dříve, než jsou nasazeny v reálných sítích. I když se NS2 potýká s některými nedostatky, které jsou popsány v práci, NS2 se jeví jako vhodným nástrojem pro simulace výkonnosti protokolu TCP.

V práci jsou zobrazeny a popsány výsledky měření pro tři nejperspektivnější vysokorychlostní varianty, HTCP, Scalable TCP a Highspeed TCP. Simulován a popsán je i dnes využívaný Reno a je srovnáván s novými vysokorychlostními variantami. Tyto nové protokoly jsou umístěny do virtuálních sítí vytvořené pomocí skriptů v prostředí Network Simulator NS2. Výsledky v této práci jsou popsány povětšinou pro ty nejhorší případy, se kterými se standardní TCP nemůže vyrovnat, tedy přenosové rychlosti Gb/s a časy RTT až 160ms.

Testovány byly i nižší přenosové pásma sdílené linky, ale výsledky vzhledem k velkému množství simulací nejsou v práci zahrnuty (jsou však na v příloze na CD). Při testování vysokorychlostních variant byl brán ohled jak na jejich výkonnost, tak i jejich férovost při vstupu na sdílené linky. Jak lze pozorovat, výkonnost nových TCP variant není tolik závislá na vyrovnávacích pamětech směrovačů. Při ztrátách paketů si dokážou poradit svým agresivním chováním, kdy zabírají po několika sekundách dostupnou šířku pásma. Takovými vlastnostmi ovšem téměř znemožní ostatní přenosy na lince, které jsou pro menší fronty směrovače úplně zahlušeny. Pokud nastavíme tyto fronty dostatečně velké (dvojnásobek BDP), přenosy se chovají férově, ale vzroste příliš jejich zpoždění. V rámci výkonnosti skončil nejlépe HTCP, v rámci férovosti HTCP, ale nedosahoval takových výsledků, které dávaly teoretické předpoklady. Dosahoval ale lepších výsledků než Scalable TCP a Highspeed TCP. Vysokorychlostní varianty jsou dostupné v jádře linuxu od verze 2.6.13.

9 LITERATURA

[1] Wu-chun Feng, Mike Fisk, Mark Gardner, Eric Weigle: Dynamic Right-Sizing: An Automated, Lightweight, and Scalable Technique for Enhancing Grid Performance

[2] Tom Dunigan, Matt Mathis, Brian Tierney: A TCP Tuning Daemon, *http://www.csm.ornl.gov/~dunigan/netperf/wad.html*

[3] SG TCP Optimizer, *http://www.speedguide.net/downloads.php*

[4] Scalable TCP, Improving performance in highspeed networks; http://www-lce.eng.cam.ac.uk/~ctk21/scalable/

[5] S. Floyd, Highspeed TCP for Large Congestion Windows, *IETF Internet draft*, *http://www.icir.org/floyd/papers/rfc3649.txt*, Srpen 2002.

.[6] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. In *Proceedings of First International Workshop on Protocols for Fast Long-Distance Networks*, Únor 2003.

[7] D. Leith, R. Shorten: H-TCP: TCP Congestion Control for High Bandwidth-Delay Product Paths, Červen 2005

[8] Marian Morávek: Ovplyvňovanie prenosových charakteristík TCP/IP komunikácie, Diplomová práca, Bratislava, Apríl 2005

[9] YongJae Chuh, JaeSub Kim, YongJoo Song and Daeyeon Park, F-TCP: Light-weight TCP for File Transfer in High Bandwidth-Delay Product, Networks, Korea Advanced Institute of Science and Technology, 2005

[10] Subodh Kerkar, Performance Analysis Of Tcp/Ip Over High Bandwidth Delay Product Networks, Final Thesis, červenec 2006

[11] František Maroušek, Varianty TCP, 2003/2004, *http://www.kiv.zcu.cz/~simekm/vyuka/pd/zapocty-2003/variantyTCP*

[12] DOSTÁLEK L., KABELOVÁ A.: Velký průvodce protokoly TCP/IP a systémem DNS, Computer Press, 2000.

[13] Ladislav Lhotka, Technologie internetu, soubor článku časopisu ComputerWorld,2007,*www.computerworld.cz/cw.nsf/id/lhotka_internet_11+tcp+window+scaling+windows+xp&hl=cs&ct=clnk&cd=3&gl=cz&lr=lang_cs*

[14] Dr. Ing. Sven Ubik, Přenosy velkých objemů dat v rozlehlých Gigabitových sítích, *www.cesnet.cz*

PŘÍLOHY

CD ROM disk, který obsahuje:

- Diplomovou práci ve formátu DOC a PDF
- Simulační prostředí Network Simulator NS2 ns-allionone- 2.31
- Patch pro vysokorychlostní varianty
- Skripty testujícího software
- Návod na sestavení a spouštění kompletního testovacího prostředí